



Water Pressure Optimisation for Leakage Management Using Deep Reinforcement Learning

By

Ahmed Mohamed Abdelkader Atia Negm

In collaboration with

Designed Network Solutions Ltd.



Designed Network Solutions

This thesis is submitted to Lancaster University for the Degree of Doctor of
Philosophy

January 2024

This project was supported by the Centre for Global Eco-Innovation and is
part financed by the European Regional Development Fund.

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ

In the name of God, the Most Gracious, the Most Merciful.

Abstract

In this thesis, we introduce a novel approach to pressure management using deep reinforcement learning (DRL) algorithms. Exploiting DRL algorithms to optimise pressure management in water distribution networks (WDNs) provides a more computationally efficient and resilient method to reduce background and burst leakage. Using DRL to manage pressure has proven as a valuable method to reduce leakage and carbon emissions in two case studies based on a real and benchmark water network. A cohort of eight DRL algorithms of varying natures are implemented on a benchmark test network and real network model of varying sizes to prove their scalability. An investigation on their ability to reduce both background and burst leakage is conducted to highlight their abilities with regards to different leak sizes.

The application of deep reinforcement learning algorithms to control leakage in WDNs builds on from two extensive reviews of leakage management and DRL applications in the urban water systems. Collating this literature pinpoints the novelty in applying deep reinforcement learning algorithms to control pressure in WDNs and provides context to the thesis. To develop DRL algorithms fit for WDN operations, a novel python-based environment is created that can communicate the hydraulic capabilities of EPANET to the DRL agent. This involved multiple design choices including action space and observation space selection as well as formulating a reward function suitable for the multiple objectives relating to leakage reduction.

Regarding background leakage, the best performing DRL algorithm resulted in 65.2% reduction in leakage in the benchmark network. However, the investigation on the real water network provided by Northumbrian Water Living has proved the strong dependency between valve locations and pressure management hence resulting in a negligible background leakage reduction. The ability of the DRL algorithms to deal with uncertainty through randomised burst nodes was investigated in the second case study. DRL policies demonstrated resilience in comparison to the standard optimisation algorithms used (differential evolution, particle swarm optimisation, and nelder mead). The best performing DRL algorithm predicted a 58.46% leakage reduction and 5650kg of reduced CO₂ emissions in the benchmark water network. On the other hand, the best DRL performance optimised the real water network by reducing the leakage by 5.79% and carbon emissions by 1999kg of CO₂.

Keywords: Leakage, pressure management, deep reinforcement learning, water distribution

Novelty and Contributions

The main contribution presented in this thesis is the application of deep reinforcement learning as a novel method to manage pressure in water distribution networks for the purpose of leakage control. Deep reinforcement learning is a recent field of research filled with novelties in every corner. Its application has prevailed in many engineering scenarios. However, the reach of DRL algorithms has not been realised fully in urban water systems. This thesis contributes to the wider research community through providing a new method to manage water distribution networks. It also includes a novel reinforcement learning environment that communicates between the DRL algorithms and the hydraulic solver. This environment is able to communicate the water network's defining properties to different DRL algorithms. It also facilitates changes dictated by the optimisation algorithms to improve the state of the WDNs.

Using the environment, we introduce the first known application of eight DRL algorithms to optimise pressure management for leakage reduction. These algorithms belong to three main DRL families (hybrid, policy driven and distributional DRL). The hybrid DRL algorithms used are Advantage Actor Critic (A2C), Deep Deterministic Policy Gradients (DDPG), Soft Actor Critic (SAC). The policy driven DRL algorithms include Augmented Random Search (ARS), Proximal Policy Optimisation (PPO), Recurrent Proximal Policy Optimisation (Recurrent PPO) and Trust Region Policy Optimisation (TRPO). Finally, we also experiment with the use of the distributional DRL algorithm; Truncated Quantile Critics (TQC). Literature suggests that this is the first approach to deploying these DRL algorithms to manage pressure in water distribution networks.

These algorithms are trained in a loop before tested under different conditions and compared to benchmark optimisation algorithms. Trained DRL algorithms can optimise water networks in real-time or near real-time depending on the network size which is a major improvement to the current practice. In addition, the proposed method incurs less data requirements and lower computational loads than the current practices of optimisation algorithms.

Publications

Journal papers, conference papers, industrial summits, posters, and presentations based on this thesis can be found below.

Journal Papers

1. Negm, A., Ma, X. and Aggidis, G. (2023a) 'Review of leakage detection in water distribution networks', IOP Conference Series: Earth and Environmental Science, 1136(1), p. 012052. Doi: 10.1088/1755-1315/1136/1/012052.
2. Negm, A., Ma, X. and Aggidis, G. (2023b) 'Deep reinforcement learning challenges and opportunities for water industry applications', Journal of Water Research. Doi: 10.1016/j.watres.2024.121145
3. Negm, A., Ma, X. and Aggidis, G. (2024) 'Deep Reinforcement Learning for Advanced Pressure Management in Water Distribution Networks', Journal of Hydroinformatics (submitted)

Conference Papers

1. Negm, A., Ma, X. and Aggidis, G. (2023c) 'Water Pressure Optimisation for Leakage Management Using Q Learning', 2023 IEEE Conference on Artificial Intelligence (CAI), pp. 270–271. Doi: 10.1109/CAI54212.2023.00120.

Industrial Summits

1. Negm, A. (2022a) 'Deep reinforcement learning for network pressure management', Global Leakage Summit Conference Presentation. Doi: 10.13140/RG.2.2.13936.12803/1
2. Negm, A. (2023d) 'Deep reinforcement learning for network pressure management', Global Leakage Summit Conference Presentation. Doi: 10.13140/RG.2.2.15433.72803

Presentations

1. Negm, A. (2021a) 'Future Water Networks', COP26 Event Lancaster University. Doi: 10.13140/RG.2.2.12448.35847
2. Negm, A. (2021b) 'Development and design of future water networks to help protect the world's most important natural resource', Lancaster University School of Engineering PGR Conference 2021. Doi: 10.13140/RG.2.2.19159.24483
3. Negm, A. (2022b) 'Deep Reinforcement Learning for Network Pressure Management', Lancaster University School of Engineering PGR Conference 2022. Doi: 10.13140/RG.2.2.16922.82880

4. Negm, A., Ma, X. and Aggidis, G. (2022c) 'Review of leakage detection in water distribution networks', IWA Hydroinformatics Conference 2023. Doi: 10.13140/RG.2.2.35746.50880
5. Negm, A. (2023e) 'Deep Reinforcement Learning for Network Pressure Management', Lancaster University School of Engineering PGR Conference 2023. Doi: 10.13140/RG.2.2.35566.38721
6. IMechE Webinar (31/01/2024): 'Application of AI in leakage management in water distribution networks'

Posters

1. Negm, A. (2021b) 'Development and Design of Future Water Networks', Lancaster University FST Week Poster Competition 2021. Doi: 10.13140/RG.2.2.13936.12803/1
2. Negm, A. (2022d) 'Water Pressure Optimisation for Leakage Management Using Deep Reinforcement Learning', Lancaster University FST Week Poster Competition 2022. Doi: 10.13140/RG.2.2.13936.12803/1
3. Negm, A. (2023f) 'Can Deep RL Save Our Water and Climate?', Lancaster University FST Week Poster Competition 2023. Doi: 10.13140/RG.2.2.13936.12803/1

Acknowledgements

I would like to thank everyone that contributed to this thesis.

To my industrial supervisor, Craig Stanners, for motivating my interest in this field and guiding me throughout the course of the PhD. Navigating the water industry with your expertise has helped me grow far beyond the scope of the research. To my academic supervisors, Professor George Aggidis and Dr Xiandong Ma, thank you for giving me the opportunity to grow as a researcher and engineer under your insightful guidance. Working with you has made, an otherwise challenging degree, smooth and enjoyable. I am forever grateful for your cheerful attitudes.

To the network analysts from Northumbrian Water Living, especially Brian Plemper, thank you for your invaluable aid obtaining WDN hydraulic files and cost-savings reports. Brian, your help explaining the networks and our discussions regarding pressure management has helped me tackle earlier dilemmas in the research.

My very dear brothers, Mo'men Negm and Saif Negm, your vibrant personalities inspire the best parts of me. Mo'men, thank you for being my backbone and my piece of home away from home and Saif, I am eager to watch and learn from you as you grow to be the capable man that we all know you for. May Allah (s.w.t) guide us to unified paths that please him and our parents.

To my lovely wife, Aisha Yusuf, thank you for being my best friend and confidant before joining me as my other half. No prayer could have prepared me for the blessing that is you. I thank Allah (s.w.t) every day and night for bringing us together in such perfect timing. Life has been magical ever since and I look forward to seeing what life has in store for us. Also, I want to thank you for introducing me to my second family, Professor Hakeem Yusuf and the incredible Mrs Karimat Yusuf. I have never felt more at home in the UK than when we visit your parents. May Allah cultivate the love and trust that we share.

My lovely parents, words fail to describe how thankful I am that I have been blessed with the two most supportive people on the planet. I have never taken one step in life that could not be attributed to your upbringing and Allah's mercy. My dear father, Mohamed Abdel Kader Negm, the pillar that supported me in every way possible. I am pleased to have a father that I can genuinely call my friend. Your guidance and advice moulded me into the man that I am today. The most beautiful gift life has to offer is my lovely mother, Lamiyaa El Kady. Thank you, mom, again and again for being an angel among us. Heaven is what I feel being around you as it lies

beneath your feet. You have been my hero and inspiration every step of the way, so I dedicate this achievement to you.

Declaration

I hereby declare that all work presented in this thesis is my own, unless otherwise cited.

Table of Contents

Abstract.....	II
Novelty and Contributions.....	III
Publications.....	IV
Journal Papers.....	IV
Conference Papers.....	IV
Industrial Summits.....	IV
Presentations.....	IV
Posters.....	V
Acknowledgements.....	VI
Declaration.....	VIII
Table of Contents.....	IX
List of Figures.....	XIV
List of Tables.....	XVI
1. Introduction.....	1
1.1. Background.....	1
1.2. Deep Reinforcement Learning for Leakage Management.....	3
1.3. Aims and Objectives.....	5
1.4. Methodology and Thesis Outline.....	5
2. Leakage Management Literature Review.....	8
2.1. Leakage Assessment.....	8
2.1.1. Top-Down Water Balance.....	12
2.1.2. Bottom-Up Water Balance.....	13
2.1.3. Infrastructure Leakage Index (ILI).....	19
2.2. Leakage Detection.....	21
2.2.1. Overview.....	21
2.2.2. Characteristics and Hydraulic Properties of Leakage.....	22
2.2.3. Classification.....	22
2.2.4. Hardware Detection Methods.....	23
2.2.5. Non-Intrusive Methods.....	26
2.2.6. Software Detection Methods.....	30
2.3. Leakage Prevention.....	37
2.3.1. Pressure Management.....	37
2.4. Summary.....	43
3. Deep Reinforcement Learning Literature Review.....	46

3.1.	Reinforcement Learning Background	46
3.1.1.	Components of RL.....	51
3.1.2.	Challenges.....	56
3.2.	Deep Reinforcement Learning.....	57
3.2.1.	Notable Deep RL Algorithms.....	57
3.2.2.	Current Trends	58
3.3.	Urban Water Systems	61
3.3.1.	Challenges and Opportunities in Urban Water Systems	62
3.3.2.	Challenges of DRL in UWS.....	64
3.4.	DRL Research in Urban Water Systems	65
3.4.1.	DRL in Water Distribution	66
3.4.2.	DRL in Stormwater Systems.....	68
3.4.3.	DRL in Wastewater Treatment	70
3.4.4.	DRL in Raw Water Treatment	72
3.5.	Future Work and Novelties.....	76
3.6.	Concluding Remarks	77
4.	Water Network – Deep Reinforcement Learning Ecosystem.....	79
4.1.	The Leakage Problem.....	79
4.1.1.	The Hydraulic Model.....	80
4.1.2.	Markov Decision Process and RL Context.....	83
4.2.	The Environment	86
4.2.1.	Wrapping and Communicating with Epanet.....	88
4.2.2.	Environment Spaces	90
4.2.3.	Step Function	91
4.2.4.	Reward Control	93
4.2.5.	Render Function.....	98
4.3.	The Agents	102
4.3.1.	Hybrid DRL Agents	103
4.3.2.	Policy Driven DRL Agents.....	109
4.3.3.	Distributional DRL Agent.....	114
4.4.	Concluding Remarks	115
5.	Background Leakage Case Study	118
5.1.	Methodology.....	118
5.1.1.	Optimisation algorithms	118
5.1.2.	Problem setup.....	119
5.1.3.	Testing.....	121

5.2.	Jowitt & Xu Network	122
5.2.1.	Results	125
5.2.2.	Discussions	131
5.3.	Northumbrian Water Network	134
5.3.1.	Results	138
5.3.2.	Discussions	142
5.4.	Concluding Remarks	144
6.	Burst Leakage Case Study	147
6.1.	Methodology	147
6.1.1.	Problem Setup	147
6.1.2.	Testing	149
6.2.	Jowitt & Xu Network	150
6.2.1.	Results	153
6.2.2.	Discussions	156
6.3.	Northumbrian Water Network	159
6.3.1.	Results	162
6.3.2.	Discussions	167
6.4.	Concluding Remarks	171
7.	Conclusions	173
7.1.	Limitations	176
7.2.	Assumptions	178
7.3.	Recommendations for Future work	179
	References	182
	Appendices	216
	Appendix A: WDN-DRL Environment Code	216
	Appendix B: Optimisation Algorithms Code	232
	Appendix C: DRL Algorithm Training Scripts	240
	TRPO	242
	PPO	244
	Recurrent PPO	247
	A2C	248
	DDPG	250
	SAC	252
	ARS	254
	TQC	255
	Appendix D: Testing Blocks (DRL and non-DRL)	259

Non-DRL algorithms.....	259
DRL Algorithms	260
Appendix E: Reward Scales Sweep	261
Appendix F: Background Leakage – Jowitt & Xu Results	265
NM	266
PSO.....	268
DE	269
TRPO.....	270
PPO.....	271
Recurrent PPO.....	272
A2C	273
DDPG	274
SAC	276
SAC Tuned.....	277
TQC.....	278
TQC Tuned.....	279
ARS	280
ARS Tuned.....	281
Appendix G: Background Leakage – SZ08 Results	283
NM	284
PSO.....	285
DE	286
TRPO.....	287
PPO.....	288
Recurrent PPO.....	289
A2C	290
DDPG	292
SAC	293
TQC.....	294
ARS	295
Appendix H: Burst Leakage – Jowitt & Xu Results	296
NM	297
PSO.....	299
DE	301
TRPO.....	302
PPO.....	304

Recurrent PPO.....	305
A2C	307
DDPG	309
SAC	311
TQC.....	312
ARS	314
Appendix I: Burst Leakage – SZ08 Results	316
NM	317
PSO.....	319
DE.....	321
TRPO.....	322
PPO.....	324
Recurrent PPO.....	325
A2C	327
DDPG	328
SAC	330
TQC.....	332
ARS	334
Appendix J: Proof of Publications	337
Journal Papers.....	337
Conference Papers.....	338
Industrial Summits	339
Presentations	341

List of Figures

Figure 2-1 Standard water balance sheet by IWA (Lambert et al., 2004)	9
Figure 2-2 Modified 24-hr leakage model based on MNF.....	14
Figure 2-3 Water-wastewater balance (Al-Washali et al., 2020)	17
Figure 2-4 The four components of leakage management policy (Liemberger and Farley, 2005)	20
Figure 2-5 Leakage detection classification.....	23
Figure 3-1 The subfields of machine learning	48
Figure 3-2 Taxonomy of reinforcement learning algorithms.	50
Figure 3-3 Standard Deep Reinforcement Learning Schematic	51
Figure 3-4 Urban Water Systems.....	61
Figure 4-1 Modifying emitter coefficient and leakage exponent in EPANET.	82
Figure 4-2 Fundamental MDP model	83
Figure 4-3 WDN-DRL ecosystem schematic	86
Figure 4-4 EPYNET wrapper communication schematic	88
Figure 4-5 Introducing leakage and action using EPYNET.	89
Figure 4-6 Step function flowchart.....	92
Figure 4-7 Episodic leakage rate vs reward scales	95
Figure 4-8 Episodic violations vs reward ratios	95
Figure 4-9 Episodic penalty vs reward scales	96
Figure 4-10 Leakage rate - reward scales 3D plot	96
Figure 4-11 Violations - reward scales 3D plot.....	97
Figure 4-12 Penalty - reward scales 3D plot.....	97
Figure 4-13 Example of the interactive map render.	98
Figure 4-14 Example of reward spread across junctions render.....	99
Figure 4-15 Example of settings render.....	100
Figure 4-16 Example of Water Loss render.	100
Figure 4-17 Example of the states render.	101
Figure 4-18 Advantage Actor Critic model schematic	104
Figure 4-19 A2C Policy Network Diagram.....	105
Figure 4-20 DDPG policy network architecture	107
Figure 4-21 SAC policy network diagram.....	109
Figure 4-22 TRPO policy network architecture	110
Figure 4-23 Recurrent PPO policy network architecture	113
Figure 4-24 TQC Policy Network.....	115
Figure 5-1 Background leakage scenario flow	121
Figure 5-2 Hyperparameter Sweep	122
Figure 5-3 Labelled Jowitt & Xu network including tanks, PRVs, nodes, and pipes.	123
Figure 5-4 Initial algorithm performance - Jowitt & Xu network	126
Figure 5-5 Initial algorithm speed - Jowitt & Xu network.....	127
Figure 5-6 Tuned algorithm performance - Jowitt & Xu network.	129
Figure 5-7 Tuned algorithm speed - Jowitt & Xu network.	130
Figure 5-8 SZ08 network architecture	135
Figure 5-9 SZ08 key visualisation of high pipe flows and nodal pressures.	135
Figure 5-10 Episodic penalty plots for different reward scales - SZ08.....	137
Figure 5-11 Reward spread across junctions.....	138
Figure 5-12 Algorithm performance - SZ08	139

Figure 5-13 Algorithm speed - SZ08	140
Figure 6-1 Labelled Jowitt & Xu network including bursts (red) tanks, PRVs, nodes, and pipes.	150
Figure 6-2 Episodic penalty for reward ratios	153
Figure 6-3 Algorithm performance - Jowitt & Xu	154
Figure 6-4 Algorithm speed - Jowitt & Xu network	155
Figure 6-5 SZ08 network architecture with bursts	161
Figure 6-6 Episodic penalty for different reward scales	161
Figure 6-7a) SZ08 nodes with pressures lower than 1m. b) Old function (blue); new tanh() function (green).	162
Figure 6-8 Algorithm performance - SZ08	164
Figure 6-9 Reward time comparison – SZ08.....	164
Figure 6-10 Algorithm speed - SZ08	166

List of Tables

Table 2-1 Summary of leakage assessment methodologies	10
Table 2-2 Pressure control actuators and uses from (Mosetlhe et al., 2020).....	39
Table 2-3 Comparison of pressure control techniques (Adedeji et al., 2018)	41
Table 3-1 Summary of reviewed articles	74
Table 4-1 Hydraulic Solvers	81
Table 4-2 Available DRL agents.....	102
Table 5-1 Benchmark pipe and node data	124
Table 5-2 Benchmark consumption factors data and reservoir levels.....	125
Table 5-3 DRL algorithm training hyperparameters.....	128
Table 5-4 Key results – Jowitt & Xu	131
Table 5-5 Summary of network parameters	135
Table 5-6 DRL agent hyperparameters - SZ08.....	137
Table 5-7 Key results - SZ08.....	142
Table 6-1 Benchmark pipe and amended node data.	152
Table 6-2 Key results - Jowitt & Xu	156
Table 6-3 Key results - SZ08.....	167

1. Introduction

Water, earth's most essential resource, dictates the health of our societies yet failures in water distribution networks (WDNs) have amounted to 51 litres of leakage per person per day in the United Kingdom (OFWAT, 2022). This amounts to 23% of all distributed water in 2022 (OFWAT, 2022). This loss of water is reflected in a loss of revenue to water companies sector wide. With the diverse nature of urban cities, rising customer demand patterns, varying landscape topologies and seasonal weather trends; managing WDNs has proved a complex task. Hence why, water companies and utilities constantly explore new avenues to incorporate and test new methods to better manage their water practices.

In this thesis, we introduce the use of deep reinforcement learning as a technique to manage pressure. The aim is to minimise the effects of background and burst leakage events without violating nodal pressure limits. We believe the use of DRL algorithms will reduce the computational load in comparison to the current standard of numerical and meta-heuristic optimisation algorithms. This could allow for the real-time control of pressure valves taking into account the uncertainties in demand patterns and randomness of leakage events.

1.1. Background

The operational management of water distribution networks (WDNs) has been a challenging task for water utilities globally as they aim to preserve valuable water and energy resources without affecting the level of customer service. The preservation of water increases in complexity as we consider the outdated infrastructure forced to keep up with the rising customer demands. The resulting network expansion often results in a heterogenous system with aging WDNs being connected to new infrastructure with different materials and age further complication WDN operations (Zaman *et al.*, 2020). External factors such as overhead loading through heavier traffic and weather fluctuations enhanced with climate change plague distribution networks further. In response to the rising challenges of water distribution in the UK, regulatory bodies such as Ofwat and the Public Accounts committee have been urging water companies to reimagine the water sector by 2050 (Mace, 2020). Main themes of the sector-wide strategy include goals to 'Deliver resilient infrastructure systems' and 'achieving net-zero carbon' that will rely on developing better water management within water distribution networks. Consequently, it is essential to improve the sustainability of water transport which has been compromised with a variety of failure incidents; the most prevalent of which are leakage and burst events. The adverse effects of leakage are not limited to the loss of capital but extends to environmental degradation in the form of greenhouse gas (GHG)

emissions, technical instability, and degradation of water quality (Al-Washali, Sharma and Kennedy, 2016).

Minimising leakage can be achieved through two methods: asset management and pressure management. Whilst asset management focuses on proactively surveying and improving the network infrastructure; pressure management handles the daily operational aspect of water distribution. The advantage of controlling water traffic through pressure valves extends beyond leakage reductions as it minimises the effect of burst events; decreases water and energy costs; and decreases carbon emissions through lower pumping needs (Rogers, 2014; Farley and Trow, 2015; Adedeji *et al.*, 2018; Negm, Ma and Aggidis, 2023a). In addition, regulating pressure fluctuations reduces asset failure rates due to transient surges or cyclic pressure (Neal Andrew Barton *et al.*, 2019). These transient surges are often consequences of network operations such as the use of fire hydrants, valve installations, heavy pumping, or flushing events. Hence, the complexity of pressure management has attracted researchers globally. There are many ways to classify the control techniques covered by the research community which are detailed in section 2.3.

The most interesting of the aforementioned techniques are those that fall into the optimisation approach. This approach employs the use of advanced optimisation algorithms to satisfy single or multiple objectives set by the user. Optimisation pressure control can be applied in large scale networks and used during minimum night flow as well as high demand periods making it an attractive option for water utilities. Likewise, due to the numerous novelties available in this approach, it has become a beacon for many researchers. However, optimisation algorithms are often handicapped with their need for data and their ability to process the input data effectively and efficiently. This flaw is poorly matched with the complexities of large WDNs which are particularly hard to manage due to numerous connections, multiple water sources, daily and seasonal variations in demand patterns, and possible pumping profiles. Tackling such high dimensional scenarios requires more extensive efforts from both industry and academia to rectify the mishandling of water distribution networks. DRL is able to minimise the need for data through the function approximation properties of their deep neural networks. It also improves on the current practices of optimisation algorithms through its ability to handle numerous variables and uncertainties in water management.

The emergence of artificial intelligence tools has re-imagined how researchers and professionals tackle multi-dimensional challenges that riddled traditional computational

techniques. The effects were substantial as artificial intelligence led advancements in many sectors such as health care (Chang, 2019; Nichols, Herbert Chan and Baker, 2019; Bullock *et al.*, 2020), engineering (Malik *et al.*, 2019), transport (Abduljabbar *et al.*, 2019), smart manufacturing (C. Li *et al.*, 2023) and many more (Luong *et al.*, 2019; Mosavi *et al.*, 2020; Zhang, Zhang and Qiu, 2020). Deep reinforcement learning (DRL, Deep RL) is an emerging field of dynamic computing that has risen through the use of deep neural networks to advance its predecessor reinforcement learning (Mnih *et al.*, 2015). Its successes rely on its applicability in real world scenarios that require learning from experience and its failures arise from challenges in instability and environment definition. The appealing nature of finding low-dimensional features that accurately represent high-dimensional real-world problems and experience driven autonomous learning makes DRL a true advancement in AI. As this field grows, researchers have developed numerous deep reinforcement learning algorithms that equip computational methods such as bootstrapping, backups, replay memory and function approximation to overcome any issues that arise and improve results (Li, 2017). In addition to numerous neural network architectures, deep reinforcement learning has quickly grown to become an unclassified jungle of artificial intelligence advancements which will be covered in section 3.1.

1.2. Deep Reinforcement Learning for Leakage Management

The thesis explores the novel deployment of several deep reinforcement learning algorithms for the operational management of water distribution network to improve the resilience of the network and leakage control in the form of advance pressure management. DRL provides a more resilient way to monitor the overarching objectives. This is necessary to model the heterogenous changing nature of water distribution networks subject to different demand patterns, weather conditions, failures, and more uncertainties. Extending this technology to the operational management of water networks is a field of untapped potential with many avenues to explore. DRL provides a method to continuously alter the model in real-time to react and adjust to the environment it is placed in. This is facilitated by redefining the pressure management as an optimisation problem and the water distribution network as a Markov Decision Process (MDP). Ultimately, this allows the WDN problem to be simplified and abstracted without losing the main parameters. MDP is based on influencing the probability of transitions between different states through actions. It is often denoted by the five tuple (S,A,P,R,γ) that stand for states (S), actions (A), probabilities/dynamics (P), reward (R) and initial state (γ) (Puterman, 1990; Desharnais *et al.*, 2004). Ultimately, MDP formalism helps evaluate sequential interaction therefore introducing the hidden time dimension which is

often overlooked in machine learning algorithms. The effects of actions in WDNs are spatial temporal hence why utilising reinforcement learning's sequential nature provides a good basis to redefining the pressure management problem. Fully developing this strategy for operational management would have monumental effects in the water industry such as.

- Minimised leakage through real-time pressure management and full utilisation of pressure valves across the network. This is achieved through smoother pressure profiles and a greater focus on pressure management during minimum night flow (MNF) hours by selecting the most appropriate valve settings for the current state of the water network.
- Lower energy consumption due to the decreased failure rates. This manifests as pumps are not required to meet the increased demands of a leaking network. Additionally, less pumping decreases costs and carbon emissions.
- Longer asset life due to decreased pumping and transient surges. Transient surges are a leading cause of pipe failures which often stem from excessive pumping.
- An adaptable approach to pressure management. DRL algorithms are highly customisable to consider new dimensions to the optimisation problem through reward formulation and hyperparameter tuning. Training loops can vary in frequency to match daily and seasonal trends hence ensuring that pressure management reflects the ground truth model accurately.

However, DRL algorithms are not without fault. Like every computational method, there are limitations and challenges to consider.

DRL algorithms can follow a bad training trajectory based on observations from the water network environment. An agent (DRL algorithm) that performs bad actions will receive bad feedback (observations) from the environment that will not help them get closer to the desired reward. This could place a stubborn agent in a spiral under the impression that there is no path for positive reward. On the other hand, an agent can exploit an action that gives them a constant positive reward instead of actively exploring the environment for better route or vice versa resulting in what is known as the exploration exploitation dilemma. Another challenge is that consequent reward can be very delayed from actions. It can be often unclear how one action can have future consequences so understanding and forecasting future value is essential.

More details on reinforcement learning (RL) and its successor DRL methods and algorithms can be found in the literature set with a particular focus on applications of DRL algorithms in urban water systems in section 3.

1.3. Aims and Objectives

The research aim is to exploit deep reinforcement learning techniques for the development of a semi-supervised, self-adaptive, real-time pressure management algorithms. Nevertheless, achieving this aim requires a set of objectives to guide the realisation of this goal.

- Produce literature sets on leakage management in water distribution networks to understand current practices.
- Produce literature sets defining the field of DRL and document any intersection between DRL and urban water systems applications.
- Develop a novel python-based environment to connect optimisation algorithms from python libraries to a hydraulic modelling software. This environment should accurately define the pressure management problem and be amenable for DRL use and regular heuristic and non-heuristic optimisation algorithms.
- Create a novel schematic to harness the power of hydraulic models within the reinforcement learning environment. This is to provide a realistic probability distribution for state transitions.
- Devise insightful data visualisation rendering functions to explain algorithms' performances for comparisons.
- Validate and test different types of deep reinforcement learning algorithms against benchmarks through model-based case studies of WDNs in their ability to minimise background leakage.
- Validate and test different types of deep reinforcement learning algorithms against benchmarks through model-based case studies of WDNs in their ability to minimise burst leakage.
- Compare the results with insights on leakage water saved and the environmental impact.

1.4. Methodology and Thesis Outline

Multiple applications of deep reinforcement learning algorithms were investigated in this thesis to validate its use as a promising alternative to current optimisation approaches for leakage minimisation through pressure management. In this section we highlight the methods used to realise the application of DRL algorithms and explain the outline of the thesis.

Literature reviews that cover and expand on current leakage management practices through critical and detailed discussions are presented in chapter two and three. Leakage management practices were classified into three main branches of leakage assessment, leakage detection and leakage prevention (or control) as detailed in (Gupta and Kulat, 2018). This is followed by a comprehensive review of deep reinforcement learning methods and algorithms that includes a novel classification tree to help navigate the field. This was contextualised by collating research of DRL deployment for UWS applications hence further highlighting all the gaps in the field of research. This creates the foundation needed to delve into the research question.

The water distribution network – deep reinforcement learning (WDN-DRL) ecosystem describes the code required to communicate effectively between the DRL algorithms and water networks which is explained thoroughly in chapter 3. This is achieved by creating models of real water networks native to a hydraulic solver software (EPANET) through SCADA, GIS files, or utility-owned data. These hydraulic models interact with DRL algorithms through a novel python-based environment and wrapping files. The techniques used to form this data pipeline include data preprocessing and cleaning, data processing and data visualisation. This modular ecosystem supports the use of foreign (non-DRL) optimisation algorithms for training and testing making it the testing ground for all optimisation methods.

Moreover, we explain the two unique case studies used to test the optimisation algorithms on a benchmarked and a real water network model in chapters 5 and 6. The DRL algorithms are tested under background leakage conditions that model undetected background leakage rates in nodes throughout the entire network in chapter 5. This is followed with experimenting under burst leakage conditions in chapter 6. In this scenario, the optimisation algorithms are subjected to major burst events with a high leakage coefficient in random nodes within the network. They are tested on their ability to counteract these events and minimise water flow through the burst nodes. The experimentation method, hyperparameters and methods used are detailed in each chapter followed by performance and speed results displayed with insightful data visualisation figures to highlight the main findings of the experiment. This will include reward comparisons and leakage comparisons between multiple DRL and non-DRL optimisation methods. Additionally, metrics eliciting the algorithms' effects on carbon emissions will highlight the effects of the pressure management techniques further revealing any additional trade-offs or relationships to be considered. Furthermore, chapters 5 and 6 will cover the critical evaluation and discussion of their results. The performance and computational efficiency of DRL will be assessed followed by contextualising the results to real-world applications and the larger research question.

The research will be concluded in chapter 7 with an overview of DRL applications demonstrated and their perceived implications in leakage reductions through advanced pressure management. The limitations associated with DRL control in WDNs are discussed in this chapter along with assumptions made. Suggestions for future research and improvements will guide the development of DRL in water distribution networks beyond the scope of this thesis.

2. Leakage Management Literature Review

This chapter details the relevant literature gathered and reviewed throughout the progress of this PhD. Initially, the research focused on identifying leakage management with the broadest lens highlighting three main sectors: assessment, detection, and prevention. The main methodologies and findings are reviewed in each sector.

The preservation of water increases in complexity as we begin to consider the outdated infrastructure forced to keep up with the rising customer demands. Network expansion often results in a heterogeneous system with aging WDNs being connected to a new infrastructure with different materials and age making the leakage problem more complicated (Zaman *et al.*, 2020). Large WDNs are particularly hard to manage due to numerous connections, multiple sources, and possible pumping profiles. For this reason, leakage management can be broadly split into three sections: leakage assessment; leakage detection & localisation; and leakage prevention (Gupta and Kulat, 2018).

2.1. Leakage Assessment

In the past two decades, leakage assessment has improved greatly by developing a more thorough understanding and modelling of water loss components. This was initiated by the stress on UK water companies to satisfy regulatory measures placed to cut leakage (Liemberger and Farley, 2005). Further advancements were introduced by exploring new approaches to water loss management (Al-Washali *et al.*, 2020). The International Water Association (IWA) were the first to develop a standard water balance sheet in 2000 (Liemberger and Farley, 2005). that has since been modified and accepted globally as a benchmark balance sheet (**Figure 2-1**) by organisations such as the Environmental Protection Agency (EPA) and American Water Works Association (AWWA).

The balance sheet highlights the difference between apparent loss (AL) and real loss (RL) that make up leakage in water networks i.e., Water loss (WL). Whilst RL is concerned with water lost in the network infrastructure such as pipes, joints, or reservoirs; AL consisted with illegal consumption and metering errors. Unbilled authorised consumption (UAC) is authorised usage without revenue which could be added to water losses to calculate the total non-revenue water (NRW). Modifications to the balance sheet were made to include different aspects to

System Input Volume	Authorized Consumption	Billed Authorized Consumption	Billed Metered Consumption	Revenue Water
			Billed Unmetered Consumption	
		Unbilled Authorized Consumption	Unbilled Metered Consumption	Non-Revenue Water
			Unbilled Unmetered Consumption	
	Water Losses	Apparent Losses	Unauthorized Consumption	
			Metering Inaccuracies and Data Handling Errors	
		Real Losses	Leakage on Transmission and/or Distribution Mains	
			Leakage and Overflows at Utility's Storage Tanks	
			Leakage on Service Connections up to Point of Customer Metering	

Figure 2-1 Standard water balance sheet by IWA (Lambert et al., 2004)

water utilities like accounting for water exported and interpreting NRW in raw water systems, treated water mains and distribution systems (Lambert *et al.*, 2004). However, the original water balance suffices for most water companies.

Leakage assessment techniques are concerned with calculating real and apparent losses. These methods can be split into two branches: Top-Down and Bottom-Up. The Bottom-Up approach includes different methodologies (minimum night flow, burst and background emissions, water and wastewater balance, infrastructure leakage index) that is covered on **Table 2-1**.

Table 2-1 Summary of leakage assessment methodologies

Approach	Advantages	Disadvantages	Requirements	Applications	References
Top-down water balance	<ul style="list-style-type: none"> • Straight forward • System Wide • Pressure independent • Less fieldwork • Globally recognised. • Cost effective 	<ul style="list-style-type: none"> • Incomplete • Different definitions for “input volume” • Overestimates RL • Inaccurate estimation of UC 	<ul style="list-style-type: none"> • Water balance sheet • Values for SIV, BC and UAC 	<ul style="list-style-type: none"> • Employed in most water utilities • Basic water loss assessment applications 	(Lambert <i>et al.</i> , 2004; Liemberger and Farley, 2005; Tsitsifli and Kanakoudis, 2010; Mutikanga, Sharma and Vairavamoorthy, 2011; Farah and Shahrour, 2017; Amoatey, Minke and Steinmetz, 2018; Bhagat <i>et al.</i> , 2019; Al-Washali <i>et al.</i> , 2020; Yu <i>et al.</i> , 2021)
MNF Analysis	<ul style="list-style-type: none"> • Real field measurements • Assessment and reduction process • Better estimate of RL 	<ul style="list-style-type: none"> • Extensive fieldwork • Pressure dependent • Less cost effective • DMA/Sector wide • Subject to data reliability issues • Inaccuracies due to annual seasonal trends 	<ul style="list-style-type: none"> • DMAs or Network zoning • Data collecting and logging equipment. • Employee training • Pressure measurements 	<ul style="list-style-type: none"> • Used for cases with DMAs that could be illustrative of the whole system. • Water loss assessment and reduction technique 	(Lambert, 2001; Alkassseh <i>et al.</i> , 2013; Farah and Shahrour, 2017; Amoatey, Minke and Steinmetz, 2018; Gupta and Kulat, 2018; Al-Washali <i>et al.</i> , 2020; Yu <i>et al.</i> , 2021)
Component Analysis (BABE)	<ul style="list-style-type: none"> • Straight forward • Analysis the subcomponents of RL • Better understanding of leakage nature • Assesses the utilities leakage response policies. • Considers network capabilities 	<ul style="list-style-type: none"> • Pressure dependent • Requires ALC • Unreliable • Underestimates RL • Broad assumptions require further calibrations (ICF) • Uses intensive network data 	<ul style="list-style-type: none"> • Pressure measurements • Local network data • Utility leakage response times • Length of mains and number of customer connections 	<ul style="list-style-type: none"> • Developed networks with ALC. • Used as a supplementary tool to investigate RL components. • Better fit for leakage reduction and management 	(Lambert, 1994; Al-Washali, Sharma and Kennedy, 2016; Amoatey, Minke and Steinmetz, 2018; Bhagat <i>et al.</i> , 2019)
Water and wastewater balance	<ul style="list-style-type: none"> • Pressure independent • Less fieldwork • Cost effective. 	<ul style="list-style-type: none"> • Focuses on RL • Needs a system with a wastewater service. 	<ul style="list-style-type: none"> • Accurate wastewater treatment plant inflows 	<ul style="list-style-type: none"> • Utilities with both water and wastewater services 	(Al-Washali, Sharma and Kennedy, 2016, 2018; Al-Washali <i>et al.</i> , 2020)

	<ul style="list-style-type: none"> • Clear assumptions • Objective UAC assumption without calculating RL 	<ul style="list-style-type: none"> • Sensitive to WWTP inflow errors. • Needs further testing. • Uncertainties from exfiltration, infiltration, and outdoor water use assumptions. • AL assumptions are unfit for developing countries. • No methodology for UAC 	<ul style="list-style-type: none"> • Estimation of UC and meter inaccuracies • Billed consumption flows. • Infiltration-exfiltration factor • Water utility data 	<ul style="list-style-type: none"> • More suitable for developed countries. • Developing countries need more accurate assumptions 	
--	----------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------	--

2.1.1. Top-Down Water Balance

A Top-Down approach is arguable the most straight forward technique to leakage assessment and is employed by most water companies and authorities. Using the IWA balance sheet (**Figure 2-1**), water utilities perform audits using collected data or informed assumptions. Ideally, the system input volume (SIV) can be found from the entire system input meter which should be checked for discrepancies. Subtracting billed authorised consumption (BC) from the input volume should yield the total non-revenue water (NRW). Iteratively, the water loss (WL) can be calculated by deducting the unbilled authorised consumption (UAC) from NRW. Meter inaccuracies should be investigated using portable flow measuring devices and accounted for in the UAC and unauthorised consumption (UC) values (Arregui, Cabrera Jr. and Cobacho, 2007; Mutikanga, Sharma and Vairavamoorthy, 2011). Unmetered UAC usually entails utility water use and firefighting. Often, companies highly overestimate their consumption to skew their audits and reduce the calculated water loss (Liemberger and Farley, 2005).

Following that, the water loss can be split into apparent losses (data handling errors and unauthorised consumptions) and real losses. In developing countries, data handling and billing errors must be accounted for and in that case historical consumption trends are extrapolated to estimate customer water use (Mutikanga, Sharma and Vairavamoorthy, 2011).

Unauthorised consumption (UC) is usually case-specific and requires transparency from the water company. Mutikanga et al. took a proactive approach by investigating billing trends and employing illegal use informers (Mutikanga, Sharma and Vairavamoorthy, 2011). Assuming the UC introduces uncertainty to the top-down approach, but it is estimated to be 0.1% or 0.25% of supplied water (SIV)(Al-Washali *et al.*, 2020). A higher value of 10% of NRW was recommended by study (Mutikanga, Sharma and Vairavamoorthy, 2011) for developing countries. These generic UC assumptions cause uncertainty and an overestimation of real losses (Gupta and Kulat, 2018). This is proved through Al-Washali et al. case studies that compared AL and RL results using different assessment methodologies (Al-Washali *et al.*, 2020).

The top-down technique is a simple, pressure independent tool that requires minimal field work making it an attractive choice to utilities globally. However, its limitations make it incomplete and is often followed with the Bottom-Up approach to better estimate AL. The development of sensor technology has proven fruitful for leakage assessment in study (Farah and Shahrour, 2017). Farah and Sahrour used a smart water system to process real-time data then deployed the water balance table and automated minimum night flow (AMNF)

measurements to accurately assess leakage. This methodology showed great promise to detect leakage events promptly and decreased NRW losses from 43% to 7% (Farah and Shahrour, 2017).

2.1.2. Bottom-Up Water Balance

The Top-Down approach is incomplete on its own, so it is commonly followed by one or more 'Bottom-up' methods. These practices are concerned with the final step of the balance sheet (hence the name) which is separating real and apparent losses. In some papers, the bottom-up approach has been synonymous to minimum night flow (MNF), but other notable methods can be used to assess the subsets of water loss.

Minimum Night Flow (MNF)

Minimum night flow analysis is currently the most popular tool in leakage assessment. The methodology is founded on the assumption that when consumption is lowest between 2:00am and 4:00am (**Figure 2-2**) (Liemberger and Farley, 2005), leakage (real loss) is at its highest. District Metered Areas (DMAs) are investigated individually for 24 hours to find the lowest flow rate i.e., the MNF. DMA is a permanently bounded, hydraulically isolated section of the network. In networks with no established DMAs, suitable areas should be temporarily isolated and recorded for the assessment across the grid (Liemberger and Farley, 2005). The isolated section must be supplied by a maximum of two input flows and monitored with 24-hour zone measurements (H2M) of inflows using portable flow sensors and logged with pressure measurements (Puust *et al.*, 2010). For intermittent supply networks, the MNF is harder to find as customers must be saturated, and tanks should be filled. Therefore, MNF could happen at any point of the day but is assumed to be in the early morning (Al-Washali *et al.*, 2020).

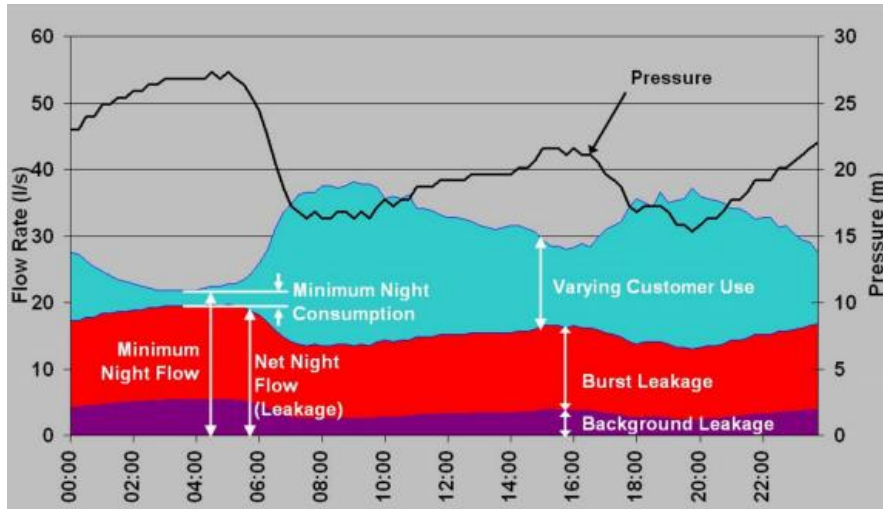


Figure 2-2 Modified 24-hr leakage model based on MNF

The next step is to calculate the legitimate night flow (LNF) and subtract it from the MNF to find the net night flow (NNF) to assess the real leakage during the MNF hour (Eq.2-1). The LNF is found by assuming that 6% of the population is active (Hamilton and Mckenzie, 2014).

$$Q_{NNF} = Q_{MNF} - Q_{LNF} \quad (2-1)$$

where Q_{NNF} is the net night flow (m^3hr^{-1}), Q_{MNF} is the minimum night flow (m^3hr^{-1}) and the Q_{LNF} is the legitimate night flow (m^3hr^{-1}).

The value of leakage calculated for NNF habitually overestimates real loss because it measures leakage at a time with increased pressure (and therefore leakage rate) as can be seen in **Figure 2-2**. Thus, a correction factor is introduced to compensate for that difference (Al-Washali *et al.*, 2020). This is achieved by applying fixed and variable area discharge path (FAVAD) principles (Lambert, 2001). Lambert dictated the essential difference in leakage rate – system pressure relationship between fixed and variable area leaks using the leakage exponential, N_1 (Lambert, 2001). For fixed area leaks this value is 0.5 and for variable area leaks it is 1.5 but since networks include a mixture of both discharge paths the exponent lies between the two. Subsequently, the value of N_1 dictates the network leaks' sensitivity to pressure fluctuations which can be used for the MNF application to form the night-day factor (NDF). NDF is essentially the sum of hourly pressures over the minimum pressure to the power of N_1 . Accounting for the NDF (**Eq. 2-2**), a more accurate value for RL flow can be obtained (**Eq. 2-3**).

$$NDF = \sum_{i=0}^{23} \left(\frac{P_i}{P_{min}} \right)^{N_1} \quad (2-2)$$

$$Q_{RL} = Q_{NNF} \times NDF \quad (2-3)$$

Where NDF is the night-day factor, Q_{RL} is the flow of real loss (m^3/hr), Q_{NNF} is the net night flow (m^3/hr), P_i is the average pressure during day (m), P_{min} is average pressure during MNF hour (m), and N_1 is the leakage exponent.

Other than providing a more reliable assessment of RL and AL (Gupta and Kulat, 2018), MNF's reliance on actual field measurements makes it eligible as a leakage reduction strategy if coupled with suitable leakage detection (Al-Washali, Sharma and Kennedy, 2016). Applications of MNF analysis for leakage reduction can be seen in Farah & Shahrour's study mentioned earlier (Farah and Shahrour, 2017) The accuracy of MNF assessment relies on the collected data and estimation issues (Al-Washali, Sharma and Kennedy, 2016) However, MNF is limited to a DMA-wide application and cannot be directly applied to the whole network. Unless the DMA or network section is monitored for the whole year, it is susceptible to inaccuracies due to consumption trends through the year (Al-Washali *et al.*, 2020). Another drawback of MNF are the intensive field work and the associated manpower (Gupta and Kulat, 2018). MNF is pressure dependent and requires a lot of field data making it less cost-effective than its counterparts (Al-Washali *et al.*, 2020).

Burst and Background Estimate (BABE)

Leakage component analysis, otherwise known as burst and background estimate (BABE), is an objective model introduced by Lambert (Lambert, 1994). It is based on the concept of multiple leakage events are the elements to real losses (Al-Washali, Sharma and Kennedy, 2016).

Subsequently, this approach determines real losses and derives the apparent losses from that. Unsurprisingly, RL subcomponent analysis is the more common application of BABE; rather than WL component analysis (Al-Washali *et al.*, 2020).

BABE is based on the logic that RL volume can be calculated from the average flow rate and run time of different individual leaks. We could categorise leakage into flow rate and duration-based types (Al-Washali, Sharma and Kennedy, 2016). Lambert (Lambert, 1994) clarifies the vast range of flow rates, which could have a high flow rate (larger than $500Lhr^{-1}$) signifying a burst or low flow rate associated with background leakages (e.g., water loss from hydrants, valves, dripping taps).

Alternatively, the duration of a leak is reflective of leakage management guidelines of the respective water company (Al-Washali, Sharma and Kennedy, 2016). Burst leakages are more

perceptible therefore can be repaired swiftly whilst background leakage run continuously undetected by most leakage detection methods. This entails the utilities' ability to effectively detect, localise and repair the bursts. Whether the burst was reported or detected through Active Leakage Control (ALC), the duration can be classified to:

1. Awareness time, concerned with the time it takes the utility to discover a leak.
2. Location time, concerned with the time it takes to correctly positioning the leak area.
3. Repair time, concerned with the time it takes to fix the leak once located (Farley *et al.*, 2008).

In order to perform the BABE approach data should be sourced from standard parameters, local network data and utility leakage procedures with respect to their authority on the duration of bursts (Lambert, 1994). The leakage volume calculation was simplified through several model assumptions founded on specific case studies (Lambert, 1994) that were enhanced over the years (Lambert, 2009). Any bursts over 500Lhr⁻¹ were considered isolated or instantly repaired.

Avoidable and Unavoidable real losses must be estimated to find the RL. The avoidable real losses can be found by from utility data on reported bursts reported by customers or detected through ALC (Farley *et al.*, 2008). Avoidable unreported real losses can be estimated using the typical flow rates represented in table 6.1 of (Farley *et al.*, 2008) at 50 metres pressure or study (Lambert *et al.*, 1999). Unavoidable RL can be estimated from its components represented in table 2 of (Lambert, 2009), number of service connections, and length of mains. The Unavoidable Annual Real Losses can be estimated using the following equation (**Eq. 2-4**) (Al-Washali *et al.*, 2020).

$$UARL = \left(\frac{18L_m}{N_c} + 0.8 + 0.025L_p \right) P_{avg}$$

(2-4)

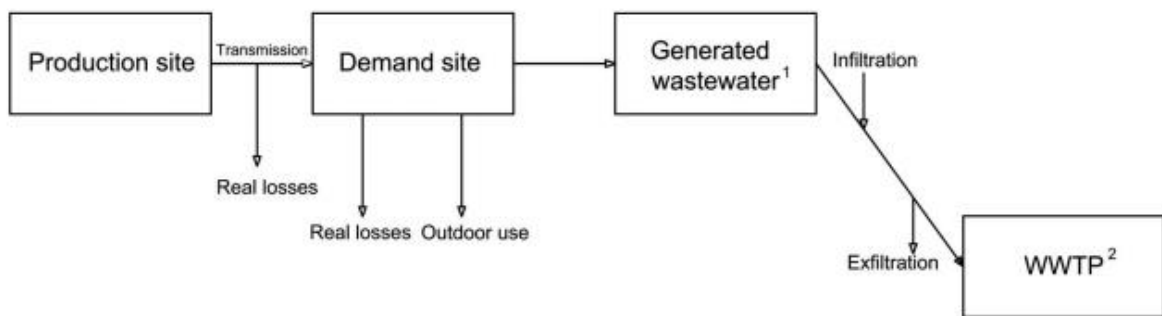
where UARL is the unavoidable annual real losses (L/service connection/day), L_m is the length of mains (L_m), N_c is the number of service connections, L_p is the average length of connection from property line to customer meter (km), and P_{avg} is the average pressure.

The BABE model exclusively identifies the RL subcomponents which could familiarise the assessors with the network's nature and leakage footprint. It also highlights the impact of the utilities' leakage policies on RL. However, the limitations to subcomponent analysis makes it unreliable for water loss assessment. The assumptions that are the foundations of the model are derived from certain cases often resulting in underestimated RL when applied to

international systems especially in developing countries (Gupta and Kulat, 2018). In an attempt to resolve that, the Infrastructure Condition Factor (ICF) ranging between 1-3 is used to correct the disparity between the model cases and real case (Al-Washali, Sharma and Kennedy, 2016). BABE can only be applied to networks that have regular ALC strategies makes it unfit for use developing countries. More limitations include pressure dependency and the need of intensive network data (Gupta and Kulat, 2018). As a result, this methodology is recommended as a supplementary tool for WL assessment but could prove more beneficial in a WL reduction and management scheme.

Water and Wastewater Balance

Water and Wastewater Balance is a novel water loss assessment method proposed by (Al-Washali, Sharma and Kennedy, 2018) based on the notion that all AL can be found from wastewater measurements. The sewer system inflow measurements are unaffected by illegal consumption or data handling errors hence provides a more accurate representation of actual water consumptions. **Figure 2-3** below illustrates the theoretical water and wastewater mass balance.



- 1: Generated wastewater from legal and illegal users, customers with malfunctioning meters, stopped meters, meters and metering with low accuracy
 2: Almost all apparent losses reach WWTP

Figure 2-3 Water-wastewater balance (Al-Washali et al., 2020)

The balance is used to derive an **equation 2-5** for Apparent Loss Estimation (ALE) where the AL volume is calculated as a function of the wastewater treatment plant (WWTP) inflow readings (Al-Washali et al., 2020). The equation compensates for the effect of outdoor use, unbilled authorised consumption (UAC) and exfiltration-infiltration using case specific factors (A, B, C).

$$Q_{AL} = (A + 1)Q_{ww} - (B - C + 1)Q_{bw}$$

(2-5)

Where Q_{AL} is the flow of apparent losses ($m^3\text{year}^{-1}$), Q_{ww} is the inflow of wastewater ($m^3\text{year}^{-1}$), Q_{bw} is the flow of billed water ($m^3\text{year}^{-1}$), A is the exfiltration-infiltration factor (3-10%), B is the UAC factor (0.5-1.5%), and C is the outdoor water use factor (4-40%).

Thus, the AL can be calculate using ALE once the factors are estimated/optimised and billed water and wastewater flows are collected. Study (Al-Washali *et al.*, 2020) uses water consumption per capita, industrial outflow and WWTP inflow to assume the exfiltration-infiltration factor, A in **equation 2-6**.

$$A = Q_{ex} - Q_{inf} = N_p \times q_{cap}(1 - C \div 100) + Q_{ind} - Q_{ww}$$

(2-6)

Where Q_{ex} is the exfiltration volume (m^3), Q_{inf} is the infiltration volume (m^3), N_p is the wastewater service population, q_{cap} is the water consumption per capita (m^3), C is the outdoor use factor, Q_{ind} is the industrial and commercial wastewater discharge (m^3), and Q_{ww} inflow of wastewater (m^3).

The unbilled authorised factor is often estimated from water utility data or 0.5% of billed water(Al-Washali *et al.*, 2020). Meanwhile, the outdoor use factor (B) can be estimated by successfully capturing the outdoor characteristics such as garden sizes, pool ownership (Arbues, Garcia-Valinas and Martinez-Espinera, 2003). Alternatively, it can be calculated using equation 2-7 the monthly billing data (Al-Washali, Sharma and Kennedy, 2018; Al-Washali *et al.*, 2020).

$$C = \frac{Q_{bc} - 12 \times q_{bc.min.month}}{Q_{bc}} \times 100$$

(2-7)

Where Q_{bc} is the volume of annual billed consumption (m^3), and $Q_{bc.min.month}$ is the volume of minimal consumption month (m^3).

ALE has two sensitivities that should be studied beforehand. The WWTP inflows should be measured during dry weathers only as rainy days will cause an overestimation of apparent losses. On rainy days, the user must discount the measurements of WWTP and replace them with the average flow of the remaining dry days (Al-Washali, Sharma and Kennedy, 2018). Secondly, the billed water and WWTP flows should be modified to only include customers with

both water and wastewater services. Once the AL have been calculated, the RL can be simply derived by subtracting the AL from the water losses (WL).

The water and wastewater balance approach introduces the first objective method to estimate apparent losses. Unlike MNF, this methodology does not require advanced metring techniques, heavy fieldwork, or even highly trained operators. Water and wastewater balance is pressure independent making it less susceptible to unreliable average pressures (Al-Washali, Sharma and Kennedy, 2018). However, the technique is limited to networks with both water and wastewater services (Al-Washali *et al.*, 2020). Also, the result is sensitive to errors in WWTP inflow uncertainties but that could be solved by introducing accurate metering equipment.

2.1.3. Infrastructure Leakage Index (ILI)

ILI's ability to efficiently capture the utilities management of real loss has made a clear PI choice for most systems. The outcome ratio is non-dimensional hence allowing direct comparisons between utilities globally. Unlike other RL performance indicators (per service connection Op28 or per km Op27), ILI (**Equation 2-8**) considers the current average pressure, customer meter location and service connection density providing a fair contrast between systems with different infrastructure characteristics (Lambert *et al.*, 2004).

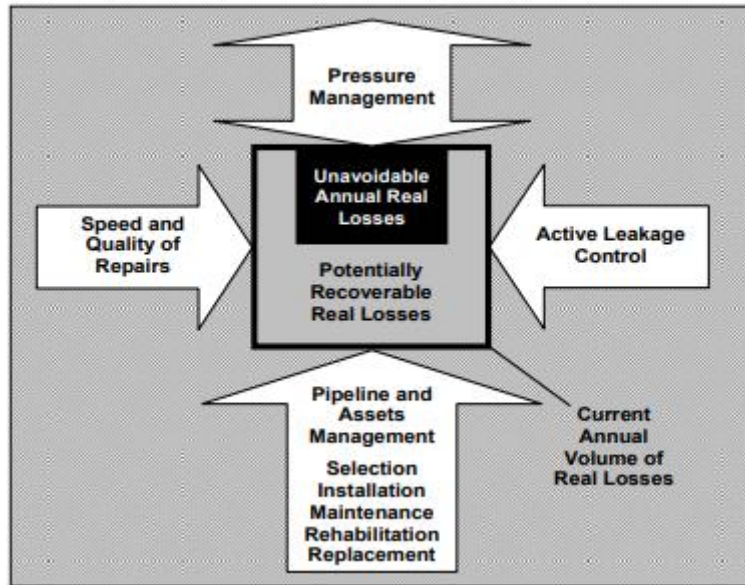
$$ILI = CARL/UARL$$

(2-8)

Where CARL is the current annual real losses and UARL is the unavoidable annual real losses (**Eq. 2-8**).

The underlying concept of ILI can be visualised clearly through the four components of leakage management shown (**Figure 2-4**). The large square represents CARL, and the inner black box is UARL.

Figure 2-4 The four components of leakage management policy (Liemberger and Farley, 2005)



The difference in their sizes represents the ILI and is a measure of the infrastructure management functions being taken (ALC, Pipeline and asset management, infrastructure repair, pressure management) (Liemberger and Farley, 2005).

Initially, UARL and ILI were developed for the component analysis of networks (BABE) and the FAVAD concept. Despite its promise and efficiency, ILI does not consider the four dimensions of sustainability (social, environmental, institutional, economic factors) and should be solely used as a technical assessor. It is not recommended for small networks with less than 5000 customers or pressure below 35 psi (Gupta and Kulat, 2018).

2.2. Leakage Detection

The search for a robust leakage detection and localisation method has been the interest of the water industry for the past two decades making it a well-developed research area. This is mainly due to the economic loss that water utilities incur through leakage and the resulting non-revenue water. On average, water networks leak 20% to 30% of the water distributed through them totalling around £7 billion of revenue loss through direct and indirect damage (El-Zahab and Zayed, 2019). The effect of leakage contributes to environment degradation through increased greenhouse gas emissions from pumping the water across the network. Contaminations from leakage often causes the water quality to decrease beyond acceptable levels thereby risking the health of the public.

Leakage detection in water distribution networks has taken many forms through investigating varying properties of leakage. Understanding the characteristic leakage types and properties introduces the different emerging technologies. Even though some methodologies have gained popularity in the past decade, the need to establish a complete, economical leakage detection solution that effectively identifies background leakage as well as burst events persists. The benefits and limitations of the aforementioned technologies has often confused water utilities on adapting the most suitable method. Therefore, there is an arising need to classify and benchmark leakage detection practices. This section reviews technology in leakage detection contrasting hardware & software, intrusive & non-intrusive, steady state & transient, single & hybrid methods. A particular focus is placed on scoping the projected direction of leakage detection and localisation. As anticipated, the various techniques refined over the last two decades introduce different capabilities, conditions, and constraints (Zaman *et al.*, 2020). Assessing and comparing those methods will provide a deeper understanding of the research area thus paving the way for novel solutions.

2.2.1. Overview

Unreported leakage can be broadly identified into two types: Burst and Background leakage (Adedeji, Kazeem B; Hamam, Yskandar; Abe, Bolanle; Abu-Mahfouz, 2017). Burst leakages are often detected through their clear properties such as acoustic emissions (AE) and significant pressure reduction (Adedeji *et al.*, 2017),(Chan, Chin and Zhong, 2018). In contrast, background leaks are often small water loss through fittings, creeping joints or small cracks which do not have inherent detectable qualities. As a result, background leakage often run for longer causing adverse losses to the network (Adedeji, Kazeem B; Hamam, Yskandar; Abe, Bolanle; Abu-Mahfouz, 2017). Across the literature, the terms burst events and burst leakage are interchangeable whilst background leakage is referred to as leakage (Chan, Chin and

Zhong, 2018). The detection of leakage can be summarised to three phases that dictate important objectives: Identify, Localise and Pinpoint (ILP) (Hamilton, 2009; El-Zahab and Zayed, 2019). The identification phase is concerned with successfully differentiating leak signals from other network signals, such as fire hydrants, to determine the presence of a leak in the network with little or no false alarms (El-Zahab and Zayed, 2019). The second phase of the ILP approach is the localisation stage. This focuses on finding the general section of the network such as a DMA (El-Abbasy *et al.*, 2016). Pinpointing attempts to site the exact location of the leak down to a radius of 20cm. Formerly, the pinpoint phase were two separate phases (locating and pinpointing, ILLP) where locating signifies estimating the leak location to a 30cm radius. The 10 cm difference makes merging the two phases logical (El-Zahab and Zayed, 2019).

2.2.2. Characteristics and Hydraulic Properties of Leakage

The detection of leaks requires a meaningful understanding of its hydraulic anatomy and detectable properties. Habitually, a leak induces a sudden pressure decline at its location that spreads through the pipes in a set of waves which could be detected through negative pressure wave (NPW) strategies (Abdulshaheed, Mustapha and Ghavamian, 2017). This pressure anomaly is difficult to detect for background leakage events and could be an indication of unaccounted demand (e.g., fire hydrants) nevertheless pressure fluctuations have been the basis of several leakage detection techniques. The inversely proportional relationship between pressure and flow rate dictates that this decrease in upstream pressure will trigger a decrease in downstream flow rates. Pressure and flow changes are the most exploited characteristics of burst events (Abdulla and Herzallah, 2015). Another measurable leak quality is the resulting acoustic emissions released by the loss of water. These vibrations display several wave properties such as reflection, refraction, absorption, and diffraction that can be exploited to identify and locate burst events (Adnan *et al.*, 2015a). These waves can be collected through a variety of sensors such as dynamic transducers, accelerometers, or microphones (Khulief *et al.*, 2012). Temperature anomalies in the vicinity of a leak arise making it another identifier that could aid in its detection (Chan, Chin and Zhong, 2018).

2.2.3. Classification

Considerable research has been taken to investigate detection strategies in water networks making it a diverse field. Therefore, it is necessary to divide these approaches into appropriate categories. A classification tree was formulated to help navigate new readers to the research area (**Figure 2-5**). The easiest discrimination of leakage detection methodologies in literature can be between hardware-based and software-based. Hardware leakage detection highlights

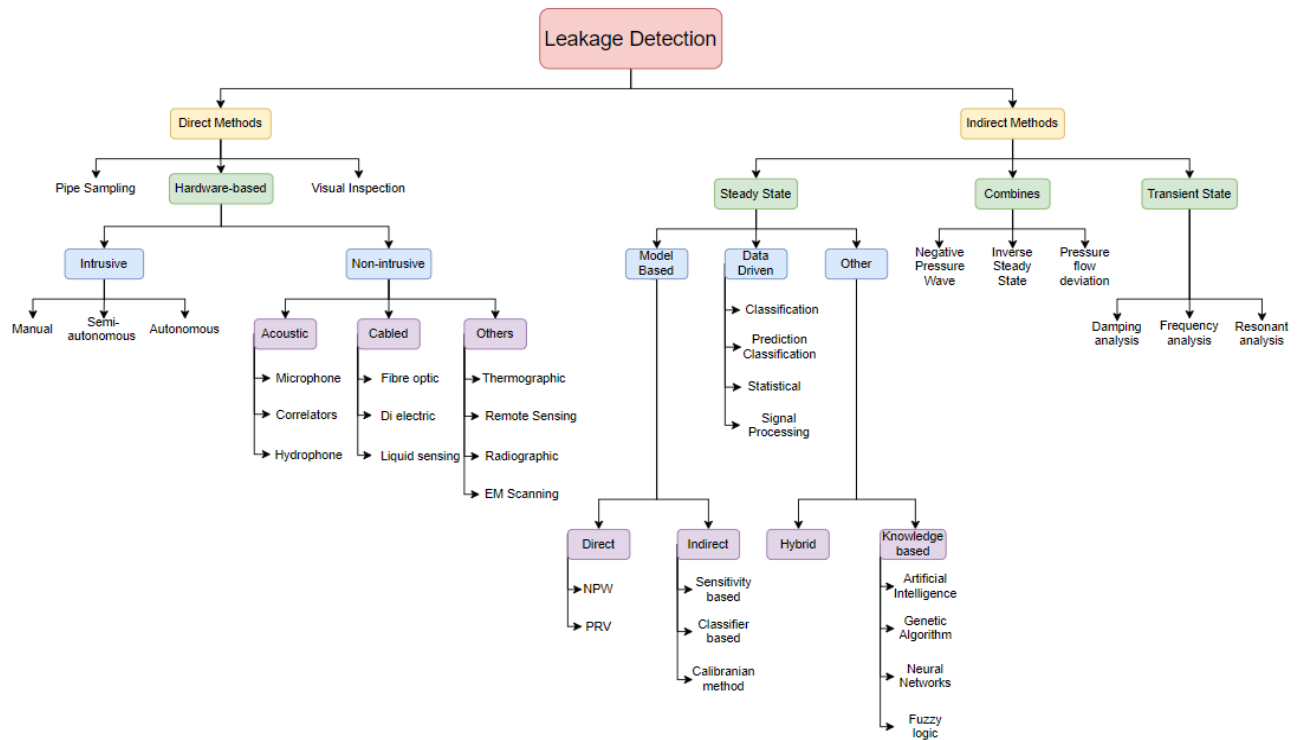


Figure 2-5 Leakage detection classification

the different sensing methods to identify and locate leakage in a network that exploit the characteristics (acoustic, pressure, flow, temperature). These can be further refined into intrusive, robotic, in-pipe systems or non-intrusive, out-of-pipe systems. However, software-based detection is more concerned with the computational and data analysis of network parameters to extract leakage information. It exceeds hardware methods in its ability to assess leakage for steady state and transient flows.

2.2.4. Hardware Detection Methods

In this section we uncover the varying hardware technology developed for leakage recognition and localisation in literature.

In-pipe Inspection Devices

Intrusive devices are an underdeveloped subsection of hardware detection methods that revolve around inspection devices that enter the pipe networks to explore leaks. Robotic inspection devices vary greatly depending on their system characteristics which include their driving method, sensing technology and level of autonomy (Tur and Garthwaite, 2010). In the following section we will review the current technologies used for each of these system characteristics including example prototypes and commercially available products. Sensing technologies will be covered later with the non-intrusive techniques.

Driving Method

Generally, moving mechanisms can be defined as passive or active where passive approaches rely on the flow of water to inspect the pipe and active approaches are equipped with their one or more actuators to achieve the desired motion. Passively propelled inspection robots are often named PIGs (pipeline inspection gauge) (Guan *et al.*, 2019). PIGs are highly effective, safe and economical devices due to their simple nature and navigation system (Tur and Garthwaite, 2010; Guan *et al.*, 2019). They have proven useful in clearing deposits gathering on the interior of the pipes in addition to their main role to assess the pipe status and detect leakages. PIG inspection systems utilise one or more sensors such as ultrasonic and eddy current sensors (Bickerstaff *et al.*, 2002). The navigation is usually achieved through odometers, visual sensors and inertial measurement units (Guan *et al.*, 2019). PIGs' motion can be problematic at higher flows where it is hard to halt their motion and when passing through corners in pipe infrastructure however intelligent PIGs might present better speed control capabilities. Commercially available PIGs include the 'smart PIG' by NORSEN GROUP; 'Smartball' by Pure Technology and 'Remoted PIG' by Jiutai Technology (Ismail, I. N.; Anuar, AS.; Sahari, 2012; Roslin *et al.*, 2012).

Active driving methods are categorised into wheel, track, inchworm, walking and snake mechanisms. Wheel propulsion is generally coupled with a spring mechanism to press against the pipe walls to smoothly adapt to the in-pipe topology. It exceeds the other driving strategies with its high efficiency, simplicity and ability for miniaturisation (Tur and Garthwaite, 2010). Wheel-based prototypes are available in the following literature (Kolesnik, Behavior and 2002, no date; Roh and Choi, 2005). Wheeled robots can also be screw-driven which usually entails a stationary and rotational section. The spiral motion transforms to linear motion through the modules (Shao *et al.*, 2015). A prototype of screw-driven wheeled robot can be found in (Shammas, Wolf and Choset, 2006). For soft or cracked ground motion, wheeled robots are often outperformed by track-driven systems. However, this drive is rarely used due to its higher complexity and energy requirements. Roman *et al.* have proposed a track-driven inspection robot in their work (Roman, Pellegrino and Sigrist, 1993). Similarly, legged robots lack the simplicity and efficiency the wheeled robots offer but they perform better overcoming obstacles like large pipe-wall deposits. An example of legged inspection robots can be found in Bradbeer *et al.*'s work (Bradbeer *et al.*, 1997). Worm-like movements are produced by the cooperation of a clamper and extensor modules to push the robot through the pipe. As the name suggests, the clamper adheres the robot to its surroundings whilst the extensor moves in the desired direction leading to a stroke motion in that direction. This motion is often used in

foreign environments where caution is a priority. Further work on inchworm movements has been conducted by (Bertetto and Ruggiu, 2001; Lim *et al.*, 2007), (Choi, Jung and Kim, 2004), (Menciassi *et al.*, 2002). Like the inchworm, snake robots have proven more adaptable in abnormal environments. They are comprised of several connected modules capable of planar movement (Liljebck *et al.*, 2012). The control of snake platforms is often challenging due to the increasing degrees of freedom with every adjoining module. Reptile movement for inspection robots is not widely used but an example can be found in Shamma et al.'s work (Shamma, Wolf and Choset, 2006).

Level of Autonomy

Intrusive hardware inspection techniques belong to one of three classes of autonomy: No autonomy, semi-autonomous, or fully autonomous. Most robotic inspection devices lie in the first category of non-autonomous hardware however the introduction of autonomy provides freedom from user interference (Tur and Garthwaite, 2010; Liu and Kleiner, 2013).

Fully operated robots are usually controlled through a tether cable by trained users or through a wireless link. The operator examines the inside of the pipe in real time using the incoming sensor data as the robot moves along the network (Tur and Garthwaite, 2010). The tether cable is preferred because it enables a smoother recovery as shown in the study (Moraleda, Ollero and Orte, 1999). Through their research, the authors concluded that there are no cost-effective solutions that can navigate the varying scenarios inside water networks (Moraleda, Ollero and Orte, 1999). This led to the popularity of tethered, non-autonomous inspection robots.

Semi-autonomous inspection is achieved through implementing automatic control modules that can remove some of the user's duties such as navigation or pipe condition assessment. This shifts some of the users' responsibilities and introduces higher accuracy. Prototypes that belong in this category include the PIRAT (Kirkham *et al.*, 2016) and Karo (Kuntze and Haffner, 1998) robots.

Fully automated robots are those voids of any user interaction. They are able to navigate, assess and communicate pipe condition through their sensor payload in real-time without being lost in the system. The challenges faced by autonomous inspection devices are numerous but the most prominent is energy and communication in long-time and long-range applications (Tur and Garthwaite, 2010). Kirchner and Hertzberg developed the Kurt robot that uses a map of the pipe network to collect video graphic, ultrasound and gradient data through automatic operation (Kirchner and Hertzberg, 1997). Similarly, the Fraunhofer Institute

produced an automated crawling inspection system, the Makro robot, equipped with additional IR sensors and laser projectors (Kolesnik, Behavior and 2002, no date; Rome *et al.*, 1999).

Prototype

In this review, we have already specified some smart PIGs and robotic detection prototypes that highlight the range of in-pipe inspection including Makro (Kolesnik, Behavior and 2002, no date; Rome *et al.*, 1999), Karo (Kuntze and Haffner, 1998), PIRAT (Kirkham *et al.*, 2016), Kurt (Kirchner and Hertzberg, 1997) and Smartball. Through the extensive efforts of the research community to find the optimal smart inspection platform, many prototypes have been created yet only a few of them have been developed into a product due to infeasibility. Finding a complete intrusive inspection robot is difficult due to the design challenges they must overcome including:

- The varying pipe diameter of the networks.
- Junctions and corners manoeuvring requires flexibility.
- Protecting the sensors from the environment.
- Multiple sensors are required to provide a more comprehensive inspection.
- Retrieval issues mean require active self-propelling devices.
- Most inspection gauges require service interruption as they empty the pipes.
- Ensuring the devices don't affect the water quality and introduce contaminants.
- Communicating a large amount of data to the network operators from a long distance.

Despite that, pipe inspection gauges can be an appealing solution to some network with successful case studies shown. Therefore, it is crucial to review intrusive robotic devices regularly to highlight any potential advancements or solutions to our modern-day leakage dilemma. The prototypes available can be found in a comprehensive summary table in (Tur and Garthwaite, 2010, p. 503) and table 2 of (Liu and Kleiner, 2013, p. 12)

2.2.5. Non-Intrusive Methods

Leakage detection systems can also be labelled as dynamic or static methods. Whilst intrusive methods are referred to as dynamic due to their motion throughout the network to investigate inner pipe conditions, non-intrusive methods depend on mounted sensors that collect data used to infer leakage making it a static method (El-Zahab and Zayed, 2019). Static methods carry the advantage of identifying a leak immediately whereas dynamic detection is often deployed after a leak is expected/identified to pinpoint the leak area (Lee *et al.*, 2005; Cataldo *et al.*, 2014). The research scene has been marginally focused on the static strategies of

leakage detection for the last two decades due to their more tangible benefits and their ability of real-time management (El-Zahab and Zayed, 2019). The most prominent technologies exploit acoustic or pressure properties accounting to more than 50% of published research (El-Zahab and Zayed, 2019)(El-Zahab and Zayed, 2019). Other technologies rely on flow sensors, ground penetrating radars (GPR), tracer gas detection, infrared thermography which will be mentioned in this section.

Acoustic Techniques

Acoustic based leakage detection and localisation can be traced back to the early 1990s in water and oil networks (Gupta and Kulat, 2018). The localisation of leak events through acoustic methods can be classified into time-of-flight-based or attenuation-based. Attenuation based relies on the decrease of signal amplitude as the acoustic signals travel across the pipeline while time based monitors the increase of signal transit time (Lee and Lee, 2000). Acoustic emissions result from turbulent pressure fluctuations at the leak, vapor bubbles forming at high velocities and imploding as shock waves on pipe walls. The frequency of those acoustic emissions (AE) varies depending on the source where turbulent flows produce low frequency signals and cavitation bursts cause high them in plastic pipes requires a denser distribution of the sensors (De Silva, Mashford and Burn, 2011). These instruments could be geophones (electrical or mechanical), hydrophones, listening sticks, accelerometers, or correlators.

Geophones are easily implemented to detect leak-induced seismic vibrations in buried pipelines (Iskander, 2018). Their ability to accurately locate leaks is aided by their high sensitivity but is often dependant on the operator's experience. Deploying geophones is often used to localise and pinpoint a pre-identified leak and hindering the area above the suspected leak unusable (El-Zahab and Zayed, 2019). Tethered and untethered hydrophones have been used as listening instruments to detect leaks. This widespread sensor is often submersed in the fluid column through hydrants and valves. Hydrophones are more accurate than geophones however they are expensive (Epa, 2010) and often lack in sensitivity for acoustic leak signals. Several studies investigated the combination of hydrophones with signal processing and cross-correlating techniques to increase sensitivity (Khulief *et al.*, 2012; Gao *et al.*, 2017). Listening probes/sticks operates as earpieces that rely on the operator's ability to distinguish the acoustics of leak. This requires highly skilled operators and low external noise which limits the effectiveness of this methodology. These devices are suited for small to medium metallic pipes with diameters between 75mm to 250mm at pressure range of 10m (Hamilton and Charalambous, 2013). The accuracy of the rods is not affected by the pipe material but is

heavily reliant on human senses. Leak noise loggers, often paired with correlators, are often used to establish a real-time leakage detection system. They are placed for long-term operations across the network with low maintenance cost but a high initial cost (Datamatic Ltd., 2008; El-Zahab and Zayed, 2019). Implementing a monitoring system in this manner relies on a communication and analysis base that can compute the incoming data allowing for faster detection and response. This aid from software and computational methods can reduce the false detection rate (Hamilton and Charalambous, 2013; El-Zahab *et al.*, 2017).

Fibre Optics

The use of optical fibres to detect and localise leakage has been adopted in water distribution system due to several benefits. It introduces a system capable of long-distance sensing with several measuring points along a single fibre hence providing accurate leak detection and localisation. These systems measure temperature anomalies inherent in leaks throughout the pipe. In comparison to oil and gas pipelines, the leak induced temperature change in water pipelines is smaller and harder to detect (Jacobsz and Jahnke, 2019). Daily and seasonal temperature fluctuations increase the difficulty of fibre optics leakage detection (Jacobsz and Jahnke, 2019). Optical fibres could alternatively monitor the strain in the pipe wall due to leaks (Inaudi and Glisic, 2008; Davila *et al.*, 2016). Developments to increase the use of fibre optics for pipeline monitoring of leak-induced temperature or strain include the use of Raman Distributed Temperature Sensor (RDTS), Fibre Bragg Grating (FBG) (Jacobsz and Jahnke, 2019) and Brillouin Optical Time Domain Reflectometry (BOTDR) (Adedeji *et al.*, 2017). The use of fibre optics exceeds other methods in its immunity to electrical noise, corrosion resistance and stability (Chan, Chin and Zhong, 2018) however their high initial and operating costs make them less desirable to water utilities. Their inability to monitor non-linear pipelines is another sign of its infancy as a leakage detection strategy. Distributed fibre optics have proven beneficial in other application but must be developed further to fit the heterogenous nature of water distribution networks.

Infrared Thermography

Like fibre optics, infrared thermography exploits the thermal effects of leakage in pipelines to identify and locate the event. IR cameras have been applied to asses pipe conditions (Gross *et al.*, 1999; Joung and Kim, 2006) as well as leakage detection in water networks (Khawandi, Daya and Chauvet, 2010; Hunaidi *et al.*, 2000). Despite it being a scarce research topic, thermography could provide a cost-effective, efficient, non-destructive way to monitor large areas of water networks. Capturing the thermal anomalies translated to the surface above the leakage is affected by several factors (Bach and Kodikara, 2017). IR should be measured during

times where the ambient temperatures is closer to equilibrium, hence increasing thermal visibility. Pre-sunrise and post-sunset hours have been suggested as the most suitable periods for thermography (Huang *et al.*, 2010; Bach and Kodikara, 2017). Soil moisture tends to hinder the investigation which can be problematic in rainy countries such as the UK despite the moist soil's superior heat transfer ability (Huang *et al.*, 2010). Ground-based thermography have to consider surface vegetation density and thermal contrasts caused by shading. Study (Khawandi, Daya and Chauvet, 2010) theorised that a 12m leak-sensor distance would be optimal for detection. Other factors that affect thermography include weather, wind, and seasonal variations. Achieving a perfect environment for IR thermography can be challenging causing varying leak thermal contrasts throughout the year. In a reliability study, Bach and Kodikra detected a discouraging 59% of the simulated leaks (Wai-Lok Lai, Dérobert and Annan, 2018). More work could be conducted in conceptualising the thermal nature of leakage and assessing the feasibility of thermography as a detecting strategy. Under the correct conditions, infrared thermography can be a useful tool for surveying leak areas, but further research and computational post processing is required for it to become a complete leakage detection strategy.

Ground Penetrating Radar

Ground penetrating radars have gained some interest in the leakage research community (Demirci *et al.*, 2012; Wai-Lok Lai, Dérobert and Annan, 2018). Their ability to utilise the electromagnetic irregularity of water leakage in infrastructure to identify and locate the failure. This imaging method excels in its applicability to both metal and plastic pipes regardless of the material and size (El-Zahab and Zayed, 2019). GPRs are easy to use and transport making it possible to survey large areas with less manpower (Hamilton and Charalambous, 2013). Despite that, the disadvantages of deploying GPRs outweigh the possible benefits. Its inability to discriminate between leak-induced irregularities and soil inhomogeneity increases the false alarm rate (Gupta and Kulat, 2018). This strategy is limited to pipes buried less than 5m deep and highly influenced by soil types. The complexity of the output data is difficult to interpret (Demirci *et al.*, 2012). In addition to that, GPR are quite expensive ranging around £10,000 to £22,000 (\$10,000 to \$31,000) (Epa, 2010; El-Zahab and Zayed, 2019). Considering these limitations and the road interruptions needed to survey pipelines; radars must be developed further. The reliability of this method can be improved through the aid of decision support systems (Kiss, Konez and Melinte, 2007) and perhaps the use of evolutionary search algorithms to obtain accurate leakage detection.

Tracer Gas

Gas injection utilises inert, non-toxic, insoluble, traceable gases such as halogens, ammonia, and helium to pinpoint leakage sites. As these gases seep through faulty infrastructure, operators detect their location by surveying the suspected area (KVS, no date). This requires a proficient knowledge of the network's flows to limit the gas flow to the suspected area by blocking other routes to exit the system. Tracer gas is able to detect both background leakage and burst events with low false alarm rate (Hunaidi *et al.*, 2000; Chan, Chin and Zhong, 2018). This method provides a simple way of detecting faulty pipelines between 75mm to 100m in diameter (El-Zahab and Zayed, 2019) regardless of the material. The fast and accurate response of this technique is crippled by its expense especially in large, low-pressure networks that require higher volumes of gas. The implementation cost of in-built sensors for monitoring and possible filtering stages makes this method unrealistic (Geiger, 2006; Chan, Chin and Zhong, 2018). The resultant environmental contamination of escaping gas makes this method more undesirable. Further research can target a conservative, economic way of deploying tracer gases.

Magnetic Induction

Magnetic induction is an accurate detection technique that establishes a communication link between two sets of sensors. One of the sets captures the flow, pressure and acoustic properties of the suspected leak from within the pipe whilst the other assesses the external factors such as humidity, temperature and soil properties outside the pipeline (Boaz, Kaijage and Sinde, 2014). Through a current-modulated signal, the coils of the magnetic transmitter induces the current to the receiver (Sun *et al.*, 2011). This communication link enables real time control of leakage detection hence increasing the response rate of utilities in harsh underground conditions. However, this strategy incurs high implementation costs making it undesirable.

2.2.6. Software Detection Methods

In this section, we explore the literature covered regarding software-based leakage detection. Background leakage poses a large threat to water networks as they often go undetected by the conventional hardware methods accumulating more losses over time. Therefore, the application of software methods is essential to counter this prevailing issue. The long-time savings that accompany software techniques are usually offset with the initial costs of installing sensors throughout the network making it a less popular option to utilities (Farley, Mounce and Boxall, 2010). Regulations placed by governments and bodies such as Ofwat encourage utilities to adopt better water and leakage management practices through

incentives and sanctions making software detection a more requested solution. These methods can be described as steady state, transient based or a combination of the two.

Water distribution systems often operate their leakage detection techniques under the premise of steady-state flow (Perez *et al.*, 2014). This method compares the behaviour of the actual network in comparison to the expected performance to detect anomalies that are often caused by leakage or blockage. An abundance of real data (historical and live) from the network suggests a data-driven approach to leakage detection however, in networks where data is scarce, a model-based approach takes precedence given that the hydraulic model is available.

Model-Based

As the name suggests, model-based detection relies heavily on the model's resemblance to the actual network, the data analysed and the arithmetic techniques employed (Zaman *et al.*, 2020). Therefore, building a realistic accurate model is crucial to the success of this technique, often having a separate calibration stage to compare the model and the network. Zaman *et al.* (Zaman *et al.*, 2020, fig. 3) display a useful general framework for the model-based leakage detection method. a used Developing a reliable replica on hydraulic simulation machines (e.g., EPANET, LOOP) should include input information of a leak-free system through different streams of information such as Supervisory Control and Acquisition (SCADA), Geographic Information Systems (GIS) and more. Some model platforms have an inherent leakage detection module such as WaterGEMS that exercises a genetic algorithm (GA) to signify potential leak nodes.

As soon as the model is complete, it is necessary to validate the model through several techniques to ensure that the model tracks the real-life example. In some cases, model pre-processing is performed before calibration to decrease the potential candidates (Perez *et al.*, 2014). The calibration techniques used are steady-state and extended period simulation (EPS) and contrast them to the field data. The indicator parameters could be the residual nodal pressures, tank water levels, roughness coefficients and they are usually compared for different scenarios to investigate any inconsistencies. The discrepancies are minimised iteratively through modifications to pipe friction factors, consumer demands, flow parameters and elevations (Sophocleous *et al.*, 2017). Once the model has been calibrated successfully, several leakage detection strategies can be applied to the model to predict and inform of possible leak locations and their corresponding sizes.

Several detection strategies have been developed for the model-based approach. They exploit the simulated parameters and field data to locate possible leak areas. However, these leakage detection models often suffer with the unaccounted ageing properties of the pipe causing the pipe diameters to decrease (Adedeji *et al.*, 2017). A simple method for investigating leakage is applying the conservation of mass calculation. Balancing the mass in and out of nodes can uncover unaccounted for loss hinting at a possible leak. Whilst this approach, works well for steady-state, they are prone to disturbances and pipeline dynamics resulting in false alarms (Wan *et al.*, 2012). A different method called pressure residual vector (PRV) exploits the leak-induced pressure changes in the real system and compares it to the leak-free model from their subsequent locations on the network (Pérez *et al.*, 2011). When the disparity between the modelled and actual pressure exceeds a pre-determined threshold, set through uncertainty analysis and statistical considerations, the area is flagged as a potential leak location and investigated (Ishido and Takahashi, 2014; Sousa *et al.*, 2014).

Indirect methods for model leakage detection can be classified into three types as shown in **figure 2-5**. Calibration-based methods rely on optimising the model calibration stage by infusing it with leakage information. This information is obtained by modelling leakage as a pressure demand. Genetic algorithms (GA) have proved as a useful evolutionary search algorithm (EAs) to investigate possible leak location through calibration (Sophocleous *et al.*, 2017). EA has been widely used in the optimisation of water distribution system design for both single objective and multi-objective as highlighted by the comprehensive literature review (Mala-Jetmarova, Sultanova and Savic, 2018). Sensitivity-based analysis is another method that exploits network models (Pérez *et al.*, 2011; Geng *et al.*, 2018) through investigating the pressure sensitivity of nodes in the model under leak and non-leak conditions. Combining the sensitivity matrix with the corresponding pressure residual vector can more accurately indicate potential leakage. This is represented in the study (Casillas, Garza-Castanon and Puig, 2013) with the aid of the angle based method. To develop that further, (Ferrandez-Gamot *et al.*, 2015) introduces a classifier-based method to detect leakage. Exploiting statistical classifiers, greatly improves fault localisation with comparison to the angle method demonstrated in (Casillas, Garza-Castanon and Puig, 2013) especially regarding demand uncertainties. Classifiers are often used as a data-driven approach however, their use in model-based detection has proved rewarding.

Data-Driven

Using abundant data, leak detection can navigate complex, heterogenous, large water distribution networks by bypassing the complications of hydraulic modelling. This makes it a

more reliable and accurate technique due to its reliance on real data at the cost of increased sensitivity to faulty sensors. These methods rely on their ability to reveal aberrant signals/patterns in the monitoring data received that could suggest the existence of a leak.

Data Pre-processing

Data-based detection often engineers one or a combination of flow, pressure, and demand readings. Consumer demand being the least probable data source can be rationalised by their uncertainty in the localisation stage (Ferrandez-Gamot *et al.*, 2015) and its relative insensitivity to smaller leak flow rates (Wu and Liu, 2017). The data used might differ in source, sample source (1-15 minutes) and length of time series which are crucial aspects to consider (Casillas, Garza-Castanon and Puig, 2013). These sensor readings are often raw and require considerable pre-processing before they can be implemented to any leakage detection algorithm. Data pre-processing often involves sorting, filtering, and transforming the incoming data making it a tedious task. Other issues such as uncertainty and variability must be considered when employing real data, yet this could be avoided in the instances data is extracted from models. Pre-processing is essential to filter erroneous data, filling gaps in the time-series and arranging the results for assessment making a critical step in data-driven leakage detection (Zaman *et al.*, 2020).

Detection

In our classification tree (**Figure 2-5**), data-driven techniques were divided into four types depending on their technical procedure. A different way to organise these techniques is to match their data source and data types. The technical procedures highlighted in this review include statistical, classification, prediction, signal processing. These methods are also used for transient leakage detection.

Pressure/Flow Monitoring

The simplest data-driven techniques utilise pressure monitoring such as negative pressure wave (NPW) and pressure point analysis (PPA). The NPW method detects the propagating pressure fluctuation at both sides of the leak through the use of transducers (Silva *et al.*, 1996). Localising the leak is established by contrasting the time difference between the reading on both sensors through cross-correlation. Applying the NPW approach practically is challenging particularly for long-range pipes (Adedeji *et al.*, 2017). Another limitation of NPW is its high false alarm rate resulting from its sensitivity to transient flows in networks. In order to increase the method's reliability, study (Tian *et al.*, 2012) proposes several improvements regarding false alarm reduction. NPW hybrid leakage detection techniques are encourage to justify the alarms such as represented in (Sun *et al.*, 2011). Pairing pressure transducers to compare their

leak results is an alternative method to reduce false alarms (Tian *et al.*, 2012). The last recommendation uses the aid of pattern recognition to distinguish leakage-induced pressure fluctuations from valve-induced variations (Tian *et al.*, 2012). Other ways to improve NPW, include implementing an adaptive threshold and improving data quality which could be achieved through filtering background noise and advanced data processing. Pressure point analysis (PPA) developed by EFA technologies Ltd. is commercially available technique that statistically analyses the mean pressure measurements along a pipe (Geiger, 2006). Similar to the other pressure-based leakage detection strategies, PPA issues an alarm when the mean pressure value drops beyond an established threshold. This method is straightforward and economic but lack credibility under transient conditions and cannot localise the leaks (Adedeji *et al.*, 2017).

Statistical Analysis

Statistical analysis techniques for leakage detection are methods that have no classification or prediction stage and depend completely on statistical theory (Wu and Liu, 2017). Statistical Process Control (SPC) is a central method in this category often using control charts to monitor measurement variations. They are also used for data pre-processing (Wu and Liu, 2017). The differences between univariate and multivariate SPC methods can be found in Jung *et al.* work (Jung *et al.*, 2015) The univariate methods elicited are Western Electric Company (WEC) rules, Cumulative Sum (CUMSUM) and Exponentially Weighted Moving Average (EWMA). WEC rules can only consider the past eight readings whilst EWMA has largest memory (Jung *et al.*, 2015). The multivariate methods described where Hotelling T^2 control chart with elliptical control and the multivariate versions of the CUMSUM and EWMA methods. Other statistical strategies include Principle and Independent Component Analysis (PCA, ICA) that are used to reduce the state space of the data without decreasing its value. ICA can be considered an extension to PCA that considers higher order statistics (Westra *et al.*, 2007). Newer methods of statistical procedures rely on data clustering (Wu *et al.*, 2016), support vector machine (SVM), artificial neural networks (ANN) (Zhou *et al.*, 2019) and newer versions of the multivariate methods mentioned earlier.

Classification Based Methods

Classification based strategies build models that can effectively distinguish (classify) normal and outlier data. The simplest form of a classification technique calculates the absolute mean difference between expected and recorded hydraulic measurements (Zaman *et al.*, 2020). More commonly used models are trained using sets of labelled normal and abnormal hydraulic data to successfully detect bursts. An example of this is the comparative study conducted by

Mounce and Machell on burst detection through flow reading analysis using static and time-delay artificial neural networks (ANN) (Mounce and Machell, 2007). The different architectures displayed a different relationship with the inputs causing improved detection due to its more dynamic nature. It is clear that the performance of the classification model relies heavily on the abundance of normal and outlier real data to train the model and the quality of the inputs used. Using a leak function and a self-organising map (SOM) ANN, the classification model can output a value from 0-1 to identify the probability of a leak at a node (Aksela, Aksela and Vahala, 2009). This method is more adept in distinguishing leak data without supervision or labelled training data (Aksela, Aksela and Vahala, 2009). The need of adequately labelled and balanced training data for both normal and outlier conditions is the main disadvantage of this technique therefore unsupervised learning is a logical step for further research. In addition to that, poorly trained classification models often lead to high false positive rates (FPR) which is another concern for classification-based leakage detection.

Prediction-classification

Unlike classification techniques, prediction-based method introduces a preliminary stage of outlier data prediction hence enabling the classification model to be built effectively with normal hydraulic data alone. An additional stage of data selection is required to achieve this which often utilises some of the statistical methods mentioned earlier (Mounce *et al.*, 2003; Mounce, Boxall and Machell, 2009). A linear Kalman Filter (LKF) could be trained using normal historical data to provide a statistical description of the current system (Ye and Fenner, 2010). This is an efficient method that can extract the prediction using live data alone. Expert systems such as Fuzzy Interference Systems (FIS) and Bayesian Interference Systems (BIS) provide reliable detection results however they can be developed further by using historical data, evolutionary algorithm (EA) and expectation maximisation (EM) to optimise their parameters. Mounce *et al.* employs a combination of an artificial neural network called mixed density network (MDN) in the prediction stage (Mounce *et al.*, 2003) followed by a FIS in the classification stage to improve burst detection in the form imitate human cognition (Mounce *et al.*, 2007). Following a prediction stage, support vector regression (SVR) was used to classify deviations in the input data for leakage detection (Mounce, Mounce and Boxall, 2011). Changes in historical data used for prediction-classification propagates the data uncertainty decreases the accuracy of leakage detection and requires a data selection stage.

Signal Processing

Digital signal processing (DSP) is a commonly used technique to improve leak detection and localisation using pressure or acoustic signals due to sharper transitions than traditional

techniques such as NPW. This benefit is often offset by the bandwidth restrictions that these methods introduce through pipe resonance (Cataldo *et al.*, 2014).

The use of time and frequency response analysis of acoustic emissions for leakage feature extractions has been fruitful leading researchers to explore hybrids that can exploit their benefits. This initiated the need for time-frequency analysis to obtain valuable information from both domains. Several types of Fourier transforms have been used to capture leakage characteristics, but short-term Fourier transform (STFT) has been the primary interest of researchers. STFT introduces a time variable to the spectrum by slicing the signal using a time window function. The short time segments (called frames) are then input to a discrete Fourier transform (DFT) to produce a time-frequency analysis. This method has been justified in multiple occasions and compared to fast Fourier transform (FFT) in (Lay-Ekuakille *et al.*, 2009) where it outperforms FFT in the uncertainty analysis. Li *et al.*'s proposed methodology of wavelet denoising and STFT combination has shown higher accuracy than other signal processing methods such as wavelet decomposition, gaussian mode, recurrence plot, Wigner-Ville distribution (WVD), Wigner-Hough Transform (WHT), and empirical mode decomposition (EMD) (Li *et al.*, no date). Fast Fourier transform has also been validated in the study (Kadri, Yaacoub and Mushtaha, 2014) through a fault detection and isolation (FDI) system of underground plastic pipes.

The use of STFT and other time-frequency analysis methods has been overshadowed by the application of wavelet transforms (WT) for leakage detection and localisation. STFT has the disadvantage of a strict resolution limit due to its limited window size which does not exist when using WT. Therefore, using this method better fits the multi-resolution nature of leakage and burst events (Wu *et al.*, 2008). WT can be used in multiple areas of signal processing such as denoising (Li *et al.*, no date), decomposition (Li *et al.*, no date), recognition, classification, and feature extraction. The effectiveness of WT relies heavily on the selection of the mother wavelet used is shifted and scaled across the signal to create daughter wavelets. This was emphasised in the work of Ahadi and Bakhtiar and their comparison of Haar and db8 mother wavelets (Ahadi and Bakhtiar, 2010). Studies (Wu *et al.*, 2008; Ahadi and Bakhtiar, 2010) further prove the benefits of WT over STFT. Wavelet transforms, however, are limited by the length of the mother wavelet and its non- adaptive nature (Adnan *et al.*, 2015a). Examples of mother wavelets used also include Meyer, Morlet, Daubechies and Mallet functions (Zaman *et al.*, 2020). WT has proven to reduce noise and better locate sharp transitions in the leak signals (Zaman *et al.*, 2020). In study (Zadkarami, Shahbazian and Salahshoor, 2017), wavelet features of pressure signals were compared and outperformed by two Multi-Layer Perceptron Neural

Networks (MLPNN) for feature extraction and leakage classification that are then fused by the Dempster-Shafer (D-S). The neural networks have outperformed the wavelet feature methodology in correct classification rate (CCR%) where D-S classifier fusion method resulted in 95.11%, wavelet at 86.94% and statistical features trailing behind at 64.56% (Zadkarami, Shahbazian and Salahshoor, 2017).

2.3. Leakage Prevention

As the third prong of the trident that is leakage management; leakage control plays an essential role in reducing the effect of leakage in water distribution networks. Leakage control is concerned by minimising the probability and magnitude of leaks by changing the operation and infrastructure. It is also called leakage prevention as its benefits extend to preventing future leakage through the smooth transport of water. Leakage control can be split into two main sections: Asset Management and Pressure Management. For the purpose of this thesis, we focus our literature review on the pressure management control strategies.

2.3.1. Pressure Management

Internal water pressure is a leading operational factor in pipe failure and therefore leakage. Hence, it is necessary to effectively reduce the service pressure of the water distribution networks to a suitable level to reduce leakage. The advantages of pressure control extends beyond leakage reductions as it increases the asset service life; decreases water and energy costs; and decreases carbon emissions (Thornton and Lambert, no date; Rogers, 2014; Farley and Trow, 2015; Adedeji *et al.*, 2018). By managing pressure effectively, we minimise the need for excessive pumping hence addressing environmental and energy concerns. The pressure management problem is multi-faceted as high pressure causes heavy loading on the pipes and increases the effects and probability of leaks whilst low pressures cause supply interruptions and disqualify utilities from meeting a minimum pressure requirement set by their regulatory bodies (e.g., OFWAT for the United Kingdom). The complexity of pressure control in WDNs increases with rising trends urbanisation and consequent rise of demands. In addition, demand pattern variations affect the pressure through the networks daily and seasonally which require a more continuous pressure monitoring. Therefore, pressure control quickly became a major research interest for everyone involved in the water industry.

Early research highlighted the strong relationship between leakage and pressure (**Eq. 2-9**) and aimed to elicit it in a proportional relationship (Lambert, 2001; Thornton, 2003; Thornton and Lambert, 2005).

$$Q_l = kP^n \quad (2-9)$$

where Q_l is the leakage flow rate (Ls^{-1}), k represents the leakage/emitter coefficient ($Ls^{-1}m^{-0.5}$), P is the pressure head (m) in the pipe while n denotes the leakage exponent. The value of n ranges from 0.5 to 2.5 depending on the type of leaks. A more comprehensive relationship was derived from the fixed area and variable area (FAVAD) concept by May (May, 1994).

$$Q_l = C_d A_l^f \sqrt{2gH} + C_d A_l^v \sqrt{2gH} \quad (2-10)$$

where Q_l denotes the leakage flow rate, C_d is the leakage discharge coefficient, A_l^f , the fixed area of leak opening, A_l^v , the variable area of leak opening. H represents the pressure head produced by pump while g is the acceleration due to gravity. Both equations are regularly used in both research and industry. They highlight the heavy involvement of pressure in leakage and are used heavily in literature. This inspired most of the pressure control work that followed it.

Actuating devices

Several network components can be utilised to achieve adequate to advanced levels of pressure control. Intelligent pressure management often requires synchronicity between pressure-influencing components to ensure that the pressure management does not lead to energy loss through head loss (Alberizzi *et al.*, 2019). Several pressure control devices are available to manage pressure in the network including pump as turbines (PAT), pressure reducing valves (PRV), pressure sustaining valves (PSV) pressure control valves (PCV) and pressure breaker valves (PBV). The most common devices are the pressure reducing valve and the pressure control valve however PATs have become a more recent focus due to the scarcity of energy and the trends to lower carbon emissions. The optimal placement of the valves and PATs have proven to be equally as important as their operation to ensure their effectiveness and minimise the operating costs (Saldarriaga and Salcedo, 2015a; Bonthuys, van Dijk and Cavazzini, 2020; Price, Abhijith and Ostfeld, 2022).

Table 2-2 Pressure control actuators and uses from (Mosetlthe et al., 2020)

Actuator	Use
Pressure reducing valves (PRV)	Regulation of pressure when and if it exceeds the set-out values
Pressure sustaining valves (PSV)	Sustain a certain specified pressure value
Pressure control valves (PCV)	Control the pressure in the identified pressure management area
Pressure breaker valve (PBV)	Force and maintain specified pressure loss across the valve
Pumps as turbines (PATs)	Regulation of pressure when and if it exceeds the set-out values and the recovery of energy.

Valve Placement

The placement of valves is crucial to experience the tangible impacts of valve operations. Otherwise, the effects of pressure control would not span across the district metred area (DMA). The placement methods can be classified broadly into three sections.

The enumerative method randomly selects areas for valve locations and their settings are optimised using optimisation algorithms. The process repeats until it reaches the best result. This method tends to be easier to apply but cannot guarantee the best result.

The pressure reference method (PRM) relies on hydraulic simulation through modellers to minimise the search space to the most suitable links. For varying demand patterns, the installation sited need to follow a rule based on a predetermined reference pressure. This rule in (Liberatore and Sechi, 2009) evaluates the difference between the input and output pressure.

$$\text{Rule: if } h_i - h_j > 0.1 \times h^{ref} \text{ pipe is selected as PRV site}$$

This rule improved on the previous standard of confirming that the input nodal pressure drops to below the reference pressure before it reaches the output node which was used in (Gupta et al., 2017). PRM is a suitable method to avoid the computational workload that comes with optimisation methods. Similar to the enumerative method; PRM cannot guarantee optimal placement but can guarantee the correct number of valves being installed.

The last section is calculus-based/optimisation methods. Studies (HINDI and HAMAM, 2007; Eck and Mevissen, 2012) minimised the PRV installations whilst minimising the pressure using mixed-integer non-linear programming (MINLP). Studies (Dai and Li, 2014; Pham, 2018) solve a localisation and control valve case using interior point optimizer (IPOPT) to use the minimum amount of valves to regulate pressure. As the evolutionary algorithms (EA) continue to dissipate into water network research; genetic algorithms (GA) were used to tackle the valve location problem in various studies (Araujo *et al.*, 2006; Nicolini and Zovatto, 2009; Nicolini, 2011). Another study (Saldarriaga and Salcedo, 2015a) employs a different EA method to better solve the resulting pareto front using a non-sorting genetic algorithm-II (NSGA-II). The use of EAs has grown in WDN literature with room to grow with the current research trends. This section introduces more advanced methods that can guarantee optimal placement and number of valves however they tend to be more computationally demanding.

Pressure Control Strategies

There are many ways to classify the control techniques covered by the research community. However, all control techniques follow one of six principles:

1. Fixed outlet pressure control (FOPC)
This method ensures that the maximum pressure entering a zone does not exceed the predetermined setting but does not adjust water pressures to meet demand variations.
2. Time-modulated pressure control (TMPC)
In this control, the outlet pressure is set to different values during the off-peak and peak durations of the day. This repeats daily and offers more flexibility than FOPC but has a poor response to sudden changes in demand requirements.
3. Flow-modulated pressure control (FMPC)
FMPC reduces the output pressure proportionally to the input flow using an additional flow-modulated controller.
4. Closed-loop pressure control (CLPC)
This method feedbacks the real-time pressure at the critical point of the zone in question (DMA) and uses that to adjust the output pressure. This provides great control of the pressure in the DMA. Yet CLPC is more expensive to implement and could increase the stress on the network elements.
5. Parameter-less P-controller
This method adjusts the pressure using the flow in a PCV making it easier to implement and respond to varying demands.

6. Optimisation approach

Using optimisation algorithms to control the network pressure has been studied extensively. This strategy is vast with diverse computational optimisation strategies.

Table 2-3 Comparison of pressure control techniques (Adedeji et al., 2018)

Method	Remarks	Cost	Limitation	Application
FOPC	Simple	Not expensive	Unable to adapt to pressure variation during peak and off-peak demands.	Used in small scale water piping networks.
TMPC	The controller used is easy to set up	A little bit expensive	Low response to water demand variations	Majorly used during the minimum night flow hours (MNFHs).
FMPC	Complex	Expensive	Low response to water demand variations	Can be used during both MNFHs and high demand period
CLPC	It provides the ultimate level of control	Expensive	There is a greater tendency for equipment failure	Can be used during both MNFHs and high demand period in real-time.
Parameter-less P-controller	The controller is easy to setup and can respond to water demand variations.	Not expensive	Practical application in large-scale water piping networks required.	Can be used during both MNFHs and high demand period in real-time.
Optimisation approach	For optimal location and opening adjustment of the pressure reducing valves	Not expensive	Practical application in large-scale water piping networks is required.	Can be used during both MNFHs and high demand period.

The most interesting of the aforementioned techniques are those that fall into the optimisation approach. This approach employs the use of advanced optimisation algorithms to

satisfy single or multiple objectives set by the user. Due to the numerous novelties available in this approach it has become a beacon for many researchers.

Optimisation strategies in pressure management of WDNs are mostly concerned with the placement and operation of valves (i.e., PRVs, TCVs). They tend to use meta-heuristic search algorithms such as genetic algorithms (GA). These methods combine beneficial properties of individual solutions in generational populations in search for the global optimum; however, they are sensitive to hyperparameter selection and incur a heavy computational processing load. This can be seen in (Gullotta *et al.*, 2021) where the authors use sequential addition (SA) and non-dominated sorting genetic algorithm (NSGA-II) to optimise valve locations and settings for a stormwater management model under water shortage conditions. Their findings highlighted the effectiveness of both algorithms and the higher computational demand of NSGA-II. Similar findings were also found in (Saldarriaga and Salcedo, 2015a) where NSGA-II was used in minimising water loss in water distribution networks. More researchers investigated the placement and operation of valves using meta-heuristic approaches such as (Araujo, Ramos and Coelho, 2006) where the authors used GA to optimal number and location of valves in addition to the optimal settings. This resulted in a decrease of leakage rates by 5.2 l/s (Araujo, Ramos and Coelho, 2006). Alternatively, the article (Mehdi and Asghar, 2019) uses the benchmark particle swarm optimisation to optimise valve settings in large scale WDNs to reduce leakage rates. As shown in the literatures, genetic algorithms and particle swarm optimisation has been the standard optimisation algorithms deployed for this problem.

A more detailed review of PM strategies can be found in (Mosetlhe *et al.*, 2020). In this review, the authors have declared three main avenues for further improvement in pressure management. Two of those studies being the deployment of emulators such as deep neural networks for the optimisation procedure and the modelling of WDNs; and the use of reinforcement learning (RL) based controllers as it bypasses the need for excessive data that often accompanies advanced optimisation algorithms. In addition, RL controllers learn from their experience with the environments which means that the accuracy of the algorithm will not be compromised through estimation of model parameters or require re-training (Mosetlhe *et al.*, 2020). Nevertheless, the use of reinforcement learning can only be truly unlocked by releasing its limits of scalability through the incorporation of deep neural networks hence the use of deep reinforcement learning (DRL). Whilst the use of DRL in the water industry is still at its infancy, there have been some applications that highlight its applicability in WDNs (Hajgató, Paál and Gyires-Tóth, 2020; Xu *et al.*, 2021; Hu *et al.*, 2023), hydro-systems (Delipetrev, Jonoski and Solomatine, 2017) and stormwater systems (Mullapudi *et al.*, 2020; Tian, Liao, Zhang, *et*

al., 2022; Z. Li *et al.*, 2023). A more recent example of the use of DRL for water industry applications can be found in (Makropoulos and Bouziotas, 2023) where the authors used agents to design off-grid water infrastructure. The only published use of reinforcement learning for pressure management of WDN was shown in (Negm, Ma and Aggidis, 2023b) where the authors deployed a simple tabular Q-learning method to highlight the feasibility of using RL as a PM strategy.

2.4. Concluding Remarks

Leakage assessment is a major component of leakage management as it identifies the whereabouts of water loss and quantifies the performance of WDNs. This section highlighted the current state of the art and potential next steps to the future of leakage assessment.

In more detail, explains the current leakage assessment methodologies; their benefits and limitations; examples from case studies; and a comparative summary table. The Top-down method has provided a benchmark for leakage assessment to define the different inputs and outputs of water in the system. Whilst this approach is widely adopted by utilities globally, it is incomplete and lacks an objective methodology to assess unauthorised consumption and data handling errors. Creating more reliable updated assumptions for UC and testing them for developing countries is a possible room for improvement. MNF analysis provides a more reliable outlook on the components of WL in the system and can be used for ALC strategies since it relies on real life measurements and not assumptions. Improving data reliability through advanced sensor technology and communications such as AMIoT or sensor fusion can improve MNF analysis and constantly monitor the level of background and burst leakage. The BABE method better describes the nature of leakage within the system through evaluating the subcomponents of RL in the system. This method has been created for developed countries and requires further validation in developing countries where apparent losses through UC and metering inaccuracies are higher. Water and wastewater balance is relatively novel however is only applicable to utilities that have a wastewater service. This is the first method to capture the apparent losses of the system through balancing the water and wastewater flow measurements. Similar to MNF, data reliability is a major concern to the methodology especially WWTP inflow measurements. The water and wastewater balance can be improved by validating it in developed countries that have less significant apparent losses [35]. Different leakage assessment approaches can be combined to offset their biases and provide additional insight on the nature of leakage in the network. The top-down approach underestimates the apparent losses whilst the water and wastewater balance overestimates AL in the system. The

different techniques can vary greatly in results therefore it is necessary to benchmark a decision matrix for choosing the most suitable approach.

On the other hand, the leakage detection field is sparse and multi-directional filled with researchers attempting to equip water companies with a solid method to tackle the heterogenous nature of this issue. Reviewing the field is challenging but this section should offer guidance to the multiple research areas and possible novelties that can be uncovered in each. Every section offers a further breakdown of the research area and a light comparison between technologies within to collate the most research findings and methodologies.

The robotic platforms reviewed include many platforms that vary in driving methods, sensing capabilities and autonomy. The active driving method sin intrusive inspection devices include wheeled, screw-driven, track-driven, worm, snake and legged showing that they all vary in applicability depending on the nature of the pipe and its environment. The autonomy level of the Smart PIGs and robots introduce an interesting trade-off between recoverability and less manpower. However, there are no economical devices capable of autonomously adjusting to all the scenarios present in all water networks and effectively communicating with the users with no intervention.

Non-intrusive hardware detection consists of a range of sensors that detect leak-induced anomalies to identify and locate events. The most common are acoustic sensors which include microphones, geophones, hydrophones, accelerometers, leak noise loggers and correlators. Other sensors mentioned are magnetic induction, infrared thermography, fibre optics, ground penetrating radar and tracer gas. These are often insufficient alone and will benefit further if paired with signal processing methods.

Software leakage detection draws from two general methodologies. Model-based leakage detection requires the analyst to construct an accurate model of the water distribution network using hydraulic analysis software and compare expected pressures/flows to the actual measurements to find outliers. In comparison, data-driven techniques include varying methods of collecting, pre-processing, and analysing data to directly find leak-induced outliers. Data-driven methods are classified into several categories based on their nature including statistical, classical, prediction and signal processing. Using models or data-driven methods rely solely on the availability of sensor data and their accuracies but can often be used together to provide a better hydraulic analysis of the system. These methods rely heavily on computational efforts to detect leakage and can benefit majorly from the rise of data engineering and artificial intelligence breakthroughs.

It is common for researchers and industry to use a hybrid between the methods reviewed to draw on their advantages and this should be explored further to use our current knowledge to bridge the faults of these methodologies. It is also crucial to explore new venues for model leakage prediction which can help identify background leakage. This can benefit from the emergence of neural networks as function approximators especially graph neural network due to their similar data types. Furthermore, neural networks models should be tested for transfer learning applications hence reducing the training time required for neural networks to model water distribution networks.

Finally, leakage control is concerned with minimising leakage effects through creating smoother pressure profiles using pressure management and minimising leakage probability through maintaining the infrastructure using asset management.

In this chapter we focus on pressure management methods for leakage control. Utilities deploy several network actuators (mostly valves) to control flow and pressure distribution throughout the network. It is adamant that valve placement is just as significant as valve management in pressure control. Nevertheless, research effort mostly prioritises the overarching control strategies. Using the optimising approach, researchers manage to introduce cheaper control strategies that are appropriate for varying WDN sizes during both MNFHs and high demand periods. The main drawback to using optimisation algorithms as a control method is its computational load, difficulties representing complex problem and inability to handle dynamic scenarios. The use of DRL algorithms shows great promise in bridging these gaps due to their capabilities in handling numerous uncertainties and reacting to dynamic environments.

3. Deep Reinforcement Learning Literature Review

Deep Reinforcement Learning (Deep RL; DRL) is an emerging field of dynamic computing that has risen through the use of deep neural networks to advance reinforcement learning (Mnih *et al.*, 2015). It has the potential to tackle complexities that used to be very challenging as it relies on deep neural networks for function approximation and representation. This technology has spread across many fields due to its impressive results and can effectively revolutionise the water industry. In this section, we explain the background of deep reinforcement learning and the milestones of this field using a novel taxonomy of the DRL algorithms. This will be followed by with a review of deep reinforcement learning applications in the water industry and will be concluded with critical insights on how DRL can benefit different aspects of the water industry.

Navigating the field of Deep RL requires a solid knowledge of its predecessor Reinforcement Learning and the major advancements that were led by the introduction of neural networks which is covered in section 3.1. A dedicated outlook on notable DRL algorithms and current research trends is covered in section 3.2. After reviewing the wider field of research, section 3.3. contextualises the deployment of DRL in urban water systems (UWS) by considering the challenges and opportunities inherent in the implementation of this novel technology. Moreover, section 3.4. focuses on a novel review of research deploying DRL in urban water systems. This in-depth review of the current research in the water industry will lead to an extensive discussion regarding the future of deep reinforcement learning in the water industry in section 3.5.

3.1. Reinforcement Learning Background

The field of machine learning (ML) has been a hot topic for researchers from diverse backgrounds such as virologist, biologists, engineers, psychiatrists, and more (Libbrecht and Noble, 2015; Nichols, Herbert Chan and Baker, 2019) due to its ability to analyse real world problems using algorithms that tackle more dynamic perspectives and improve with experience (Shinde and Shah, 2018). Machine learning begun as researchers hoped to achieve a novel area where instrumentation can achieve innate learning and demonstrate more 'intelligent' behaviour. From the first ML algorithm in 1951 named 'response learning algorithm' until the current day, artificial intelligence has only been empowered by this new field (Shinde and Shah, 2018). Some of the major achievements in ML was the creation of the algorithms Linear Classifier, Naive Bayes, Bayesian Network, Support Vector Machines (SVM), k-Nearest Neighbour (k-NN) and Artificial Neural Networks (ANN) (Shinde and Shah, 2018). ANNs were then adapted further to introduce deep layer and hence the introduction of Deep Learning.

ML has successfully developed the world of artificial intelligence into a true hope for near-human intelligence. Machine learning methods are often split into supervised learning or unsupervised learning methods. Where supervised learning depends on our prior knowledge and labelled examples to form an understanding of the model; unsupervised learning aims to learn some hidden structure using feature extraction of the unlabelled dataset. Supervised learning methods are more commonly used in classification and regression problems such as object detection and rainfall prediction models (Shinde and Shah, 2018; Nichols, Herbert Chan and Baker, 2019). Unsupervised learning is more equipped to tackle clustering, association analysis and feature engineering (Libbrecht and Noble, 2015). Whilst both forms of learning have greatly advanced their respective fields and widened the scope of artificial intelligence; they fall victim to the curse of time. As time passes, the models built using typical ML approaches become more and more outdated and require retraining using newer and more relevant data. Overlooking the sequential nature of engineering applications such as water distribution management can have grave consequences when implementing ML models. An example of that can be the effects of annual seasonality and age on the pipe failure frequency. Hence, the need to develop a learning approach that incorporates the hidden dimension of time – Reinforcement Learning. **Figure 3-1** highlights the place of RL as a subfield of machine learning. RL's ability to consider the effects of time through semi-supervised learning was the first expression of artificial foresight in machine learning and its closest form to human intelligence.

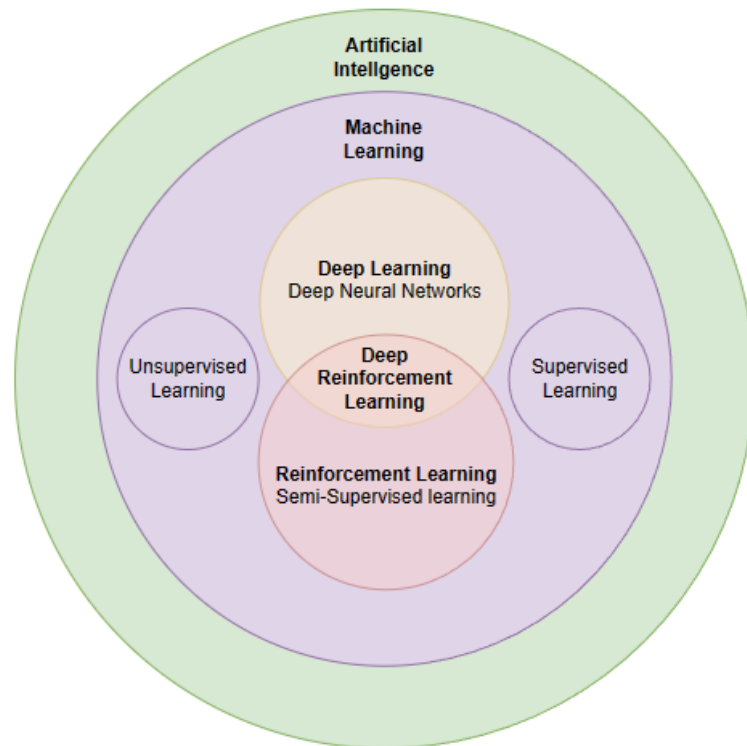


Figure 3-1 The subfields of machine learning

In its infancy, the use of reinforcement learning (RL) was an exciting concept that promised an introduction to responsive and continuously-learning AI systems. A behaviourist mathematical approach for experience-driven learning was finally attainable through RL (Sutton and Barto, 2018). This entails a reward-driven learning from interaction with an unmapped environment rather than hard computing or supervised learning where it is near difficult to obtain examples of desirable behaviour. Despite the initial successes of RL (Tesau and Tesau, 1995; Singh *et al.*, 2002; Kohl and Stone, 2004), it could not escape the ‘curse of dimensionality’ when applied to real life problems. RL was limited by complexity issues ranging from memory complexity, computational complexity and sample complexity (Strehl *et al.*, 2006).

The recent surge of deep learning and deep neural networks that has spearheaded the movement in function approximation and representation learning giving hope to unlock the true potential of RL by overcoming the issues of scalability; hence the rise of the field of deep reinforcement learning (DRL, Deep RL). This is demonstrated as the overlap between reinforcement learning and deep learning in figure 2-1. The first breakthrough use of neural networks in reinforcement learning was in Mnih *et al.*’s study (Mnih *et al.*, 2013) in which convolution neural networks were used for value function approximation. This was developed to form the basis of the first DRL method; the deep Q-networks (DQN) (Mnih *et al.*, 2015).

As deep reinforcement learning gained popularity and developed further, the field of reinforcement learning was quickly populated with novel algorithms. The field of RL has quickly transformed to a forest of methods, architectures and concepts that are difficult to navigate because of its non-modularity. To highlight the diversity in RL, we have gathered and classified a novel taxonomy of the algorithms (**Figure 3-2**). This classification tree can serve as a map to highlight the place of our algorithms in the field of DRL. It classifies the algorithms based on model free vs model based; on policy vs off policy; value-based vs policy-based; gradient based vs gradient free labels. Dotted lines are used to label dynamic programming, monte carlo, temporal difference and distributional RL algorithms. In the following sub-sections, we define the main labels used in the classification tree to elicit a better comprehension of the RL landscape.

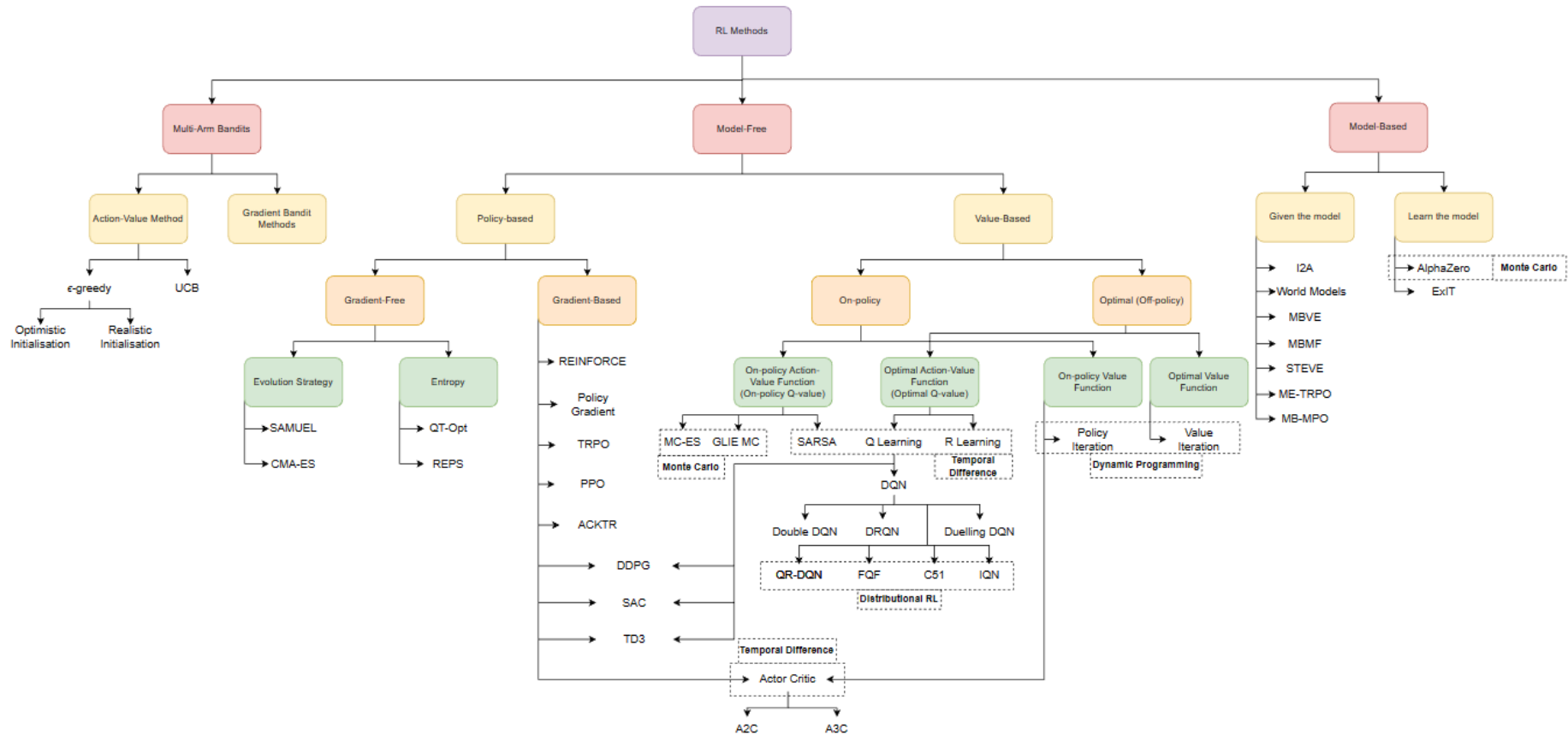


Figure 3-2 Taxonomy of reinforcement learning algorithms.

3.1.1. Components of RL

To fully comprehend the aspects and range of methods available in deep RL, it is crucial to delve into the formalism that make the RL paradigm. Reinforcement learning tackles its problems as Markov Decision Processes (MDPs) which is a commonly used description in the field of computing that depict real world processes. MDP formalism is based on evaluating the probability of transitions between different states in its process and is sometimes denoted with the five tuple (S,A,P,R,γ) that stand for states(S), actions (A), probabilities/dynamics (P), reward (R) and initial state (γ) (Puterman, 1990; Desharnais *et al.*, 2004). This helps evaluate the sequential interactions between actuators (agents, A) and their environment to influence both the state of the agent (state, S) and the relevant state of the environment (observation). The agent is then fed the observation data and a reward signal (Reward, R) that serves as an assessor to the new state that this action has led to. The aim of the agent is to find the optimal policy (π) that will maximise the expected reward which is achieved by learning the probability of state transitions attached to a state-action pair. A visual description of this process can be found in **Figure 3-3**. The deep neural network is an addition only found in Deep RL methods whilst RL methods tend to use a tabular data frame. The components of RL and DRL can be therefore redefined to suit most real-world applications in an organic and straightforward manner.

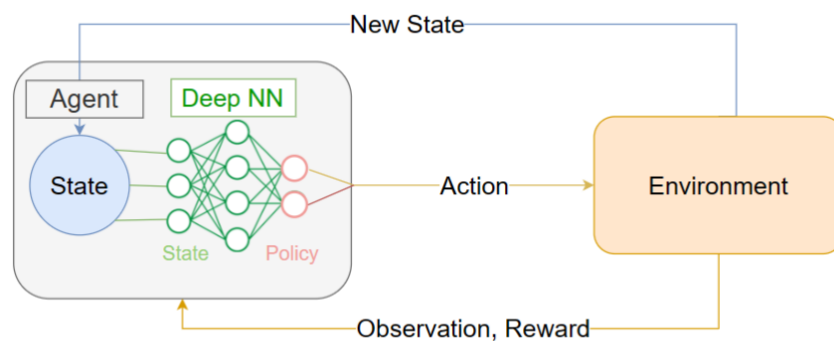


Figure 3-3 Standard Deep Reinforcement Learning Schematic

Reward and Return

The reward signal (r) is the crucial identifier that tells the agent whether their action was beneficial or harmful. The cumulative reward over a trajectory is named the return ($R(\tau)$) and it can be a finite-horizon undiscounted return (Eq. 3-1) or an infinite-horizon discounted return (Eq. 3-2). Finite return is the sum of rewards for a fixed number of steps whilst infinite returns, like the name suggests, is the summation of the sum of all the rewards ever. The infinite

returns must include the discount factor $\gamma \in (0,1)$ used to control how much weight should be placed on the agent's foresight. This helps the infinite sum converge to a finite value.

$$R(\tau) = \sum_{t=0}^T r_t. \text{ For finite-horizon undiscounted return.} \quad (3-1)$$

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \text{ For infinite-horizon discounted return.} \quad (3-2)$$

This return is usually modified and incorporated into a value function for value-based RL methods or an objective function for policy-based RL methods. Both methods have their advantages and disadvantages; for example policy-based methods are generally less sample efficient than Value based algorithms but can learn stochastic policies and converge faster than their alternative (Lapan, 2019). We discuss this further in the classifiers section below.

Value Functions

Value functions are used in almost every RL algorithm. They are a fundamental concept in RL which calculates the expected infinite horizon return to evaluate how beneficial individual states or state-action pairs are. Value functions that solely evaluate the current state without the action are often denoted by the symbol $V(s)$ and named state value functions (**Eq. 3-3**). Alternatively, state-action value functions are called quality functions, and they provide more of an insight on the trajectory of the agent given its current state-action pair (**Eq. 3-4**). The Q-value is denoted by the symbol $Q(s,a)$.

$$V(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid S_t = s\right] \quad (3-3)$$

$$Q(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid S_t = s, A_t = a\right] \quad (3-4)$$

Where $\mathbb{E}[\cdot]$ is the expected discounted infinite horizon return, s is the state sampled from S_t , a is the action sampled from A_t and t is any time step.

An important property of RL is foresight which enables agents to weight the future consequences of their actions using the expected return hence it is rare to find value functions operating without the incorporation of the bellman equations (Bellman, 1952). Bellman equations are self-consistency equations integral to dynamic programming and MDPs that follow the concept that the value of any starting point is the reward you expect from being at the starting point in addition to the value of the next point (Bellman, 1952; Puterman, 1990). Because the actions taken by an agent depend on the policy that it follows, value functions are often described in relation to its policy. On-policy value functions estimate the expected returns as the agent follows the behavioural policy (π). On-policy value functions can either evaluate a state (state-value function) or a state-action pair (state-action value function or

quality function). On-policy state-value functions are denoted by $V^\pi(s)$ and evaluates the expected return as the agent acts under behaviour policy (π) and starts with state (s) and is followed by the state (s'). This is described using the following equation (**Eq. 3-5**):

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+2+k} \mid S_t = s] = \mathbb{E}_\pi[r(s, a) + \gamma V^\pi(s')] \quad (3-5)$$

The bellman equation decomposes the value function to the sum of the current value and the future discounted values. Similarly the Q-value denoted by ($Q^\pi(s,a)$) bellman equation is formally defined as the expected return as the agent acts under the behavioural policy (π) starting with the state-action pair (s,a) and followed by the next state-action pair(s',a') (**Eq. 3-6**):

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+2+k} \mid S_t = s, A_t = a] = \mathbb{E}_\pi[r(s, a) + Q^\pi(s', a')] \quad (3-6)$$

When attempting to find the optimal policy and action for a RL problem, off-policy value functions are used to remove the restrictions of the behavioural policy and allow the agent to explore the value function following the optimal policy This leads to the off-policy state value function and off-policy state-action function. These are also called the optimal value functions ($V^*(s)$ and $Q^*(s,a)$). The main difference between the on-policy and optimal bellman equations is that the optimal uses the maximum rewardable action as shown in the equations below (**Eq. 3-7, Eq. 3-8**).

$$V^*(s) = \max_a \mathbb{E}[r(s, a) + \gamma V^*(s')] \quad (3-7)$$

$$Q^*(s, a) = \mathbb{E}[r(s, a) + \gamma \max_{a'} Q^*(s', a')] \quad (3-8)$$

The optimal action of an RL problem can be extracted by finding the maximum reward argument of the off-policy state-action value function bellman equation (optimal Q-function). In instances where there are multiple optimal actions, the algorithms often select an action at random (Achiam, 2020). Another method to evaluate the value of an action is by using the advantage function ($A(s,a)$). This compares how beneficial an action is to the average value of all actions by subtracting the state value from the state-action value under policy (π) (**Eq. 3-9**).

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (3-9)$$

The use of advantage function is intuitive as it evaluates the performance of actions relative to an average. It is simpler to compare the consequence of an action with respect to another. Learning the advantage, rather than the quality or state function, has been a recent trend in DRL algorithms (Schulman *et al.*, 2015; Wang *et al.*, 2015; Gu *et al.*, 2016; Mnih *et al.*, 2016)

For more details on the basics of value functions, we recommend the following introductory books, papers and articles (Arulkumaran *et al.*, 2017; Li, 2017; Sutton and Barto, 2018; Achiam, 2020).

Policy Driven

Other than value-based algorithms, there are policy driven techniques to solve the reinforcement learning problem and reach an optimal policy. Whilst the value-based methods use a learnt value functions to reach an implicit policy, policy-based methods do not use a value function but directly learns a policy. The value function approach often works well but it is important to be aware of its limitations. Value functions' approach to policy optimisation is focused mostly on deterministic policies which is rare in the real world since optimal policies are often stochastic. They also are subject to high sensitivities as a minor change in the expected value of an action might cause the algorithm to accept or reject it. This has been identified as a key fault that inhibits the convergence of value-based methods such as Q learning, SARSA and dynamic programming methods (Baird, 1995; Gordon, 1995; Bertsekas, Tsitsiklis and Τσιτσικλής, 1996). Policy driven methods bypass these limitations leading to better convergence properties, ability to learn stochastic policies hence more effective algorithms for higher dimensional and continuous action spaces. However, these methods can habitually converge to local minimums and are more computationally demanding with higher variance.

Direct policy search methods fine tune a vector of parameters (θ) to select the best action to take for policy $\pi(a | s, \theta)$. The policy π_θ is updated to find the maximum expected return. They can either employ gradient free or gradient based optimisation. Gradient free algorithms often use the concepts of evolution strategies (Gomez and Schmidhuber, 2005; Koutník *et al.*, 2013; Salimans *et al.*, 2017) or the cross entropy function (Kalashnikov *et al.*, 2018). Gradient-free optimisation methods can perform well in low dimensional spaces and update non-differentiable policies but, despite some successes in applying them to neural networks, the favoured method remains gradient-based training for DRL algorithms. Gradient based training methods are more sample efficient when dealing with high parameter policies (Arulkumaran *et al.*, 2017).

The gradient-based policy methods, also called policy gradient, optimise a selected objective function ($J(\pi_\theta)$) which can be defined by the average reward formulation or start-state formulation (Sutton *et al.*, 2000) simplified below (**Eq. 3-10**). Policy function approximation is challenging since gradients cannot be used through samples of a stochastic function hence

why use a gradient estimator; the theory of the REINFORCE algorithm (Williams, 1988, 1992; Sutton *et al.*, 2000).

$$J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^T R(\tau) ; \pi_\theta] = \sum_{t=0}^T P(\tau; \theta)R(\tau) \quad (3-10)$$

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \sum_{t=0}^T P(\tau; \theta)R(\tau) \quad (3-11)$$

The objective function (J) of the parameterised policy (π_θ) is the expected average return (R) under trajectory (τ). The trajectory is defined by parameterised policy.

The aim is to optimise the policy through gradient ascent by numerically defining the gradient of policy performance ($\nabla_\theta J(\pi_\theta)$) also called the policy gradient (**Eq. 3-11**). A full derivation of the policy gradient can be shown in (Achiam, 2020) however the policy gradient can be redefined as (**Eq. 3-12**).

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t)R(\tau)] \quad (3-12)$$

Where the policy gradient is the expected sum of returns (R(τ)) multiplied by the gradient of the log of the parameterise policy ($\nabla_\theta \log(\pi_\theta(a_t | s_t))$) for timesteps (t) in episode length (T). This is the simplest policy gradient; there are different variations of the policy gradient definition like the Expected Grad-Log-Prob Lemma (EGLP Lemma) (Schulman *et al.*, 2015; Achiam, 2020).

Policy-based and value-based RL coincide at the actor-critic algorithm (A2C) where the actor performs and action using policy-based RL, and the critic evaluates the resulting reward using a value function. The critic influences the actor using temporal difference error (TD error) to improve the algorithm's performance.

Other RL Algorithm Terminology

To fully comprehend the algorithms covered in the next section, it is necessary to explain the parlance and methods that form those algorithms. One way to describe RL algorithms is whether the agent is provided with a state transition function (model-based) or having to learn solely from experience through trial and error (model-free). Agents that have access to a model make use of sample efficiency and display a heightened ability of foresight but can often underperform when applied in real-world applications due to discrepancies between the model used for training and the ground-truth model. Model free methods can be implemented and easily tuned to real world application (Li, 2017). Algorithms can also be trained on sequentially generated data (online mode) or on a pre-set training batch (offline mode).

A commonly used label for RL methods is whether it is on-policy or off-policy. On policy methods evaluate or improve the behavioural policy of the current action-value pair of the current policy (e.g. SARSA) whilst off-policy methods explore the best value policy without necessarily following the current behavioural policy; they are also called optimal methods (e.g. Q-learning) (Arulkumaran *et al.*, 2017; Li, 2017). The value functions used to achieve were highlighted previously.

3.1.2. Challenges

Building deep RL algorithms is a science. In this section we build on the challenges and trade-offs underlined in the previous sections inherent in algorithm design. It is crucial to note that the field of RL research, much like the algorithms, has been expanded by experience followed by theory. In essence, some challenges were identified but not completely understood such as the deadly triad issue (Sutton and Barto, 2018).

In RL algorithm design, most researchers will make use of some form of function approximation, bootstrapping or off-policy. Function approximation uses examples to generalise an entire function hence it aids with the scalability and generalisation issue that riddles tabular algorithms and is the main tide driving the success of deep neural networks in reinforcement learning (DRL). On the other hand, bootstrapping used in DP and TD fields help with improving the algorithm's data efficiency, hence reducing computational loads. Finally, off-policy methods free our agent from target policy to explore optimality. Separately, each of these methods help RL researchers reach their desired benefits and design a better optimisation algorithms, however when combined the same methods induce instability and divergence – the deadly triad issue (Tsitsiklis and Van Roy, 1997; Sutton and Barto, 2018).

Another common challenge is the 'credit assignment problem'. This refers to the notable phenomena of incorrectly evaluating the credit of the action due to unclear or unforeseeable consequences manifesting later (Arulkumaran *et al.*, 2017). These long-term dependencies are necessary to allow the agent to better comprehend the value of its action. Hence, value functions have been modified to incorporate the estimated subsequent rewards and they have been discounted to signify the dwindling nature of consequence.

Finally, the exploration versus exploitation dilemma. This problem riddles most RL (and DRL) algorithms as agents tend to behave in a reward greedy manner. Since the agent's observation depends on its actions and its actions depend on the reward generated; RL agents can find themselves in a loop around a local optimum rather than finding the global optima - exploitation. Ultimately, the only way to solve this is to introduce randomness to the agent's

behaviour hence allowing the agent to receive new observations and possibly lead it to the global optima – exploration. This trade-off in agent behaviour has been navigated in many ways and the simplest is the use of ϵ -greedy exploration policy where the agent acts randomly with probability $\epsilon \in [0,1]$. The value of ϵ decreases as time passes leading the agent to a more exploitative nature as it learns. For continuous control, more complex methods have been used to introduce randomness over time to preserve momentum (Lillicrap *et al.*, 2016; Arulkumaran *et al.*, 2017). Other methods to tackle the exploration-exploitation dilemma include Osband *et al.*'s bootstrapped DQN using experience replay memory (Osband *et al.*, 2016), Usunier *et al.*'s exploration in policy space (Usunier *et al.*, 2017) and upper confidence bounds (UCB) (Lai and Robbins, 1985; Arulkumaran *et al.*, 2017; Pathak *et al.*, 2017).

These challenges are inherent in most RL problems and navigating them is a skill necessary to develop an effective RL algorithm.

3.2. Deep Reinforcement Learning

Many successes have stemmed from scaling RL using deep neural networks through function approximation. Deep neural networks can be used to approximate the optimal policy (π^*) or the optimal value functions (Q^* , V^* , A^*). In this section, we discuss the current trends and notable deep reinforcement learning algorithms that have progressed the field. This will help contextualise the current state of the research field and expose any future work.

3.2.1. Notable Deep RL Algorithms

The timeline and milestones that led to the creation of DRL was well illustrated in (Nguyen, Nguyen and Nahavandi, 2020, fig. 1) showing how trial and error learning, TD learning and deep neural networks came together to incentivise the first deep reinforcement learning algorithm – the deep Q-network (DQN). DQN was first introduced by Mnih *et al.* as they used convolutional neural networks (CNN) to feature engineer images from a series of 49 games (Mnih *et al.*, 2015). It was then used to tackle MuJoCo physics problems (Duan *et al.*, 2016) and three-dimensional maze problems (Beattie *et al.*, 2016). Following the success of DQN, researchers have built on the existing DQN architecture to improve its performance hence creating new algorithms such as Double DQN (DDQN) and Duelling DQN (D-DQN). Double DQN minimises the effect of noise on DQN by avoiding the overestimation of Q values (Van Hasselt, Guez and Silver, 2016) whilst the duelling network architecture combines two streams of data (the value stream and advantage stream) to produce a more accurate Q function (Wang *et al.*, 2015).

Another milestone was the introduction of the Actor-Critic algorithms that combine the use of value functions and policy gradients to forego the trade-off of variance reduction in policy methods and bias introduction from value functions (Konda and Tsitsiklis, 1999; Schulman *et al.*, 2015). Quickly, the DRL research community has direct their efforts to improve the AC methods. Schulman *et al.* (Schulman *et al.*, 2015) improves the actor using generalised advantage estimation (GAE) to produce better variance reduction baselines. The critic is also improved separately using target network in (Mnih *et al.*, 2015). Introducing deterministic policy gradients (DPG) in actor-critic algorithms was first observed in (Silver *et al.*, 2014). DPGs allow the use of policy gradients in deterministic policies when they were initially exclusive to stochastic policies. This lowers the computational load as DPGs only integrate over the state space and can therefore tackle large action spaces using less sampling. Stochastic Value Gradients (SVG) are another method to apply standard gradients to stochastic policies by ‘reparametrizing’ (Kingma and Welling, 2013; Rezende, Mohamed and Wierstra, 2014). This trend was first introduced in (Heess *et al.*, 2015) and created a flexible method capable of being using with and without value function critics and models (Arulkumaran *et al.*, 2017). SVG and DPG provide algorithmic means of improving learning efficiency in DRL.

On the lines of learning efficiency, Google’s DeepMind lab released the Asynchronous Advantage Actor Critic algorithm(A3C) (Mnih *et al.*, 2016). This advancement entails the use of an advantage function in an actor-critic architecture through training parallel agents asynchronously and aggregating their learning using a separate agent (global network). This method yields high accuracy and is applicable in continuous and discrete action spaces (Lapan, 2019) hence creating a trend for asynchronous and parallel learning. An example of A3C, and subsequently target-driven RL, in robotic navigation was demonstrated by Zhu *et al.* (Zhu *et al.*, 2016) to find the minimum sequence of actions leading to a target location using RGB images as an input.

3.2.2. Current Trends

The field of DRL is growing exponentially as researchers ground their understanding of reinforcement learning in human psychology. Using methods that parallel our natural learning trends has helped develop DRL methods further leading to fields such as hierarchical reinforcement learning (HRL) and inverse reinforcement learning (IRL). Moreover, there is more effort on improving algorithms by modelling the reward as a distribution of values similar to our brain’s reward system (Dabney *et al.*, 2020). Multi agent reinforcement learning (MARL) models the real-world nature of multiple agents interacting with the same environment and reward probability. In this section of the review, we focus on current trends in the field of deep

reinforcement learning. We explain the recent advancements and highlight notable work and challenges that are being addressed.

Hierarchical Reinforcement Learning

As the field of DRL grows, researchers have learnt how to include biases into the algorithm's learning experience. Hierarchical reinforcement learning (HRL) is a field of DRL dedicated to introducing inductive biases by factorising the final policy into several levels through state or temporal abstractions. This approach allows algorithms to tackle higher and lower level goals simultaneously by allowing top-level policies to focus on the main goal and sub-policies to focus on fine control (Tessler *et al.*, 2017; Vezhnevets *et al.*, 2017). This is how HRL attempts to achieve compositionality; achieving new representations by the combination of primitives (Hutsebaut-Buysse, Mets and Latré, 2022). The challenges faced in HRL stem from the selection of sub-behaviours or policies and how to efficiently learn state abstractions.

Inverse Reinforcement Learning

As humans, we can often learn from others' mistakes and successes. Similarly, researchers have developed methods to bootstrap the learning process using trajectories from other controllers. This is known as imitation learning (also known as behavioural cloning). The success of behavioural cloning led to the success of an autonomous car using ALVINN in (Pomerleau, 1989). The main challenge with imitation learning is its susceptibility to uncertainties. Imitation learning's inability to adapt can lead the agent down a destructive trajectory hence why it is paired with reinforcement learning. Using RL, the policy can fine-tune whilst imitation learning guides the general learning leading to faster convergence properties and better stability properties. Introducing behavioural imitation to DRL births the field of inverse reinforcement learning (IRL). IRL applies behavioural cloning by relying on provided trajectories for the desired solution to approximate the reward function (Ng and Russell, 2000). Intuitively, the motivation behind using IRL usually includes learning behaviour from experts, assisting humans and learning about systems (Adams, Cody and Beling, 2022). Application of IRL are mostly concerned with teaching robots to imitate experts (Adams, Cody and Beling, 2022). Notable work and algorithms in this field include (Ziebart and Fox, 2010; Finn, Levine and Abbeel, 2016; Ho and Ermon, 2016; Levine and Van De Panne, 2018; Paine *et al.*, 2018; Peng *et al.*, 2018).

Distributional Reinforcement Learning

Distributional RL grounds itself in our natural brain reward system (Dabney *et al.*, 2020). Like our natural dopamine system, DRL displays returns as a value probability distribution learned

from interacting with the environment. This parallel between distributional RL and our brains opens up opportunities for collaboration between AI and neuroscience (Lowet *et al.*, 2020). This new method of value distribution has shown its usefulness in improving learning speed and stability. The original distributional reinforcement learning algorithm is the categorical DQN (C51) (Bellemare, Dabney and Munos, 2017) where using value distributions the authors have surpassed most gains on the Atari2600 environment thus beating the benchmark DQN and DDQN. Other algorithms include quantile regression DQN (QR-DQN) which uses quantile regression to minimise the Wasserstein metric and improve greatly on the previous C51 in the Atari 2600 (Dabney *et al.*, 2017). Implicit quantile regression (IQR) and fully parameterised quantile function (FQF) are the latest algorithms in distributional RL and they build further on the foundations of QR-DQN (Dabney *et al.*, 2018; Yang *et al.*, 2019).

Multi Agent Reinforcement Learning

With the rising complexity of real-world systems, deep reinforcement learning algorithms often play catch-up to be able to process and scale their models. Most of the methods devised for DRL algorithms aim to simplify complex environments and feature extraction. On the other hand, multi agent deep RL introduces complexity in its algorithms by introducing several agents in the algorithms that simultaneously interact with the environment. This is representative of having multiple employees working as a team to carry out a desired goal (or policy) on the same system. The complexity of the algorithms brings forth multiple challenges that are currently the focus of the research community with the promise to solve more complex environments and real-world problems. There have been different approaches to tackle MADRL including sending signals to the agents, having bidirectional channels between the agents and an all-to-all channel (Arulkumaran *et al.*, 2017). Major challenges in the field stem from non-stationarity, partial observability, complexity in training schemes, application in continuous action spaces and transfer learning (Nguyen, Nguyen and Nahavandi, 2020). Previous reviews and surveys include (Nguyen, Nguyen and Nahavandi, 2020) that provides a review of MADRL challenges, solutions, applications and perspectives; (Buşoniu, Babuška and De Schutter, 2008) evaluates stability and a taxonomy of MADRL algorithms; (Bloembergen *et al.*, 2015) surveys dynamical models devised for multi agent systems; (Hernandez-Leal, Kartal and Taylor, 2019) bridges the gap between DRL and MADRL including benchmarks for MADRL. Other notable reviews include (Da Silva, Taylor and Costa, 2018; Hernandez-Leal, Kartal and Taylor, 2018).

3.3. Urban Water Systems

Urban water systems are a collection of complex infrastructure and processes that supply, treat, transport, and manage water and wastewater within urban environments. These systems are crucial for managing the supply of clean drinking water as well as treating wastewater and controlling storm water. Hence, they are paramount for the sustainability and well-being of cities. Effective management of UWS through sustainable practice aims to ensure a resilient supply of clean water despite climate change and seasonality. It should also minimise water loss through leakage and energy consumption through inefficient water supply and distribution. The key processes in UWS can be split into four major systems which are raw water treatment plants, water distribution networks, wastewater treatment plants, and stormwater systems (Loubet *et al.*, 2014; Etikala, Madhav and Somagouni, 2022) . Some of the processes involved in each function are displayed below in **Figure 3-4**.

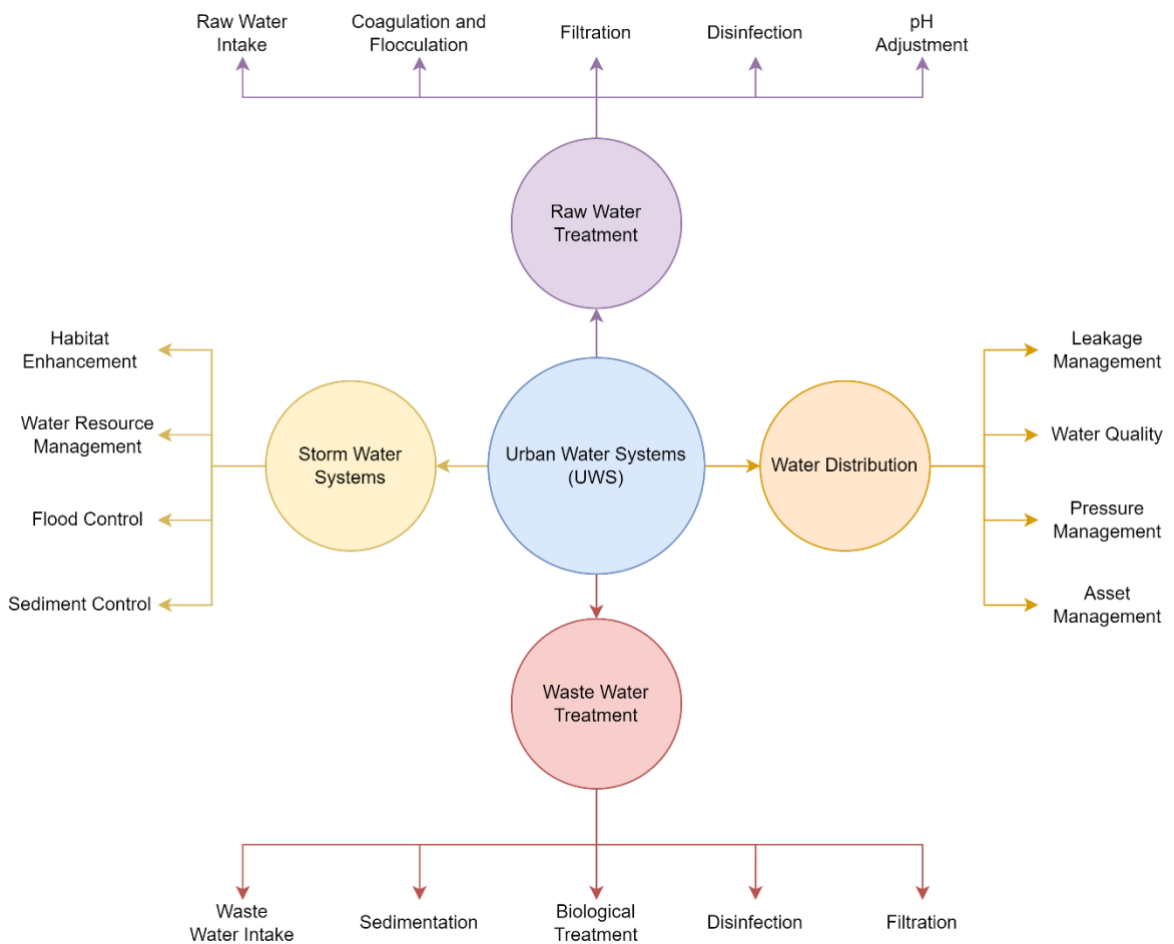


Figure 3-4 Urban Water Systems

Urban areas often obtain their water from several resources such as rivers, lakes, groundwater, and desalination plants which are managed by raw water treatment plants. Raw

water goes through several treatment processes to remove impurities, and contaminants. The main treatment methods used in raw water treatment plants include screening through mesh filters or screens, coagulation, flocculation, sedimentation, filtration, disinfection, corrosion control, pH adjustment, fluoridation, and quality monitoring (Benjamin, 2014; Jiang, 2015; Teodosiu *et al.*, 2018; Lipps, Braun-Howland and Baxter, 2022).

Once treated, clean water is distributed from the plants to the customers through a network of pipes, valves, pumps, and reservoirs. This process requires advanced pressure and asset management to minimise leakage and contamination. Due to the varying elevations, demand and climate change, the distribution of water increases in complexity and leakage has become a natural phenomenon in water distribution networks (Xu *et al.*, 2014; Neal Andrew Barton *et al.*, 2019).

Similar to raw water treatment, wastewater treatment plants are concerned with treating wastewater collected through a sewer pipeline network. Treatments include a variety of physical and chemical processes. Physical methods of screening, grit removal, sedimentation, and filtration remove heavier contaminants and large contaminants. Water is then treated biologically in the secondary treatment by using microorganisms to break down organic matter in wastewater (Hussain *et al.*, 2021). Coagulant and flocculants help remove fine particles and dissolved contaminants during the tertiary advanced chemical treatment. A final step of disinfection could use chemicals such as chlorine and UV to remove harmful pathogens (Kentish and Stevens, 2001; Crini and Lichtfouse, 2019).

During detrimental events such as floods and storms, stormwater management controls the impact on the environment and infrastructure (Ahiablame and Shakya, 2016; Aryal *et al.*, 2016; Jefferson *et al.*, 2017). Stormwater management deal with several high-level objectives such as flood control, water quality monitoring, erosion/sediment control, groundwater recharge (Jotte, Raspati and Azrague, 2017).

3.3.1. Challenges and Opportunities in Urban Water Systems

UWS include a wide range of processes that are riddled with unique dependencies and impacting factors. However, the preservation and use of water is a holistic process that incorporates the wider ecosystem, climate, and wildlife as much as human use.

Understandably, UWS share challenges that stem from external factors and opportunities to adapt deep reinforcement learning techniques. In this section, common current challenges that plague UWS processes are discussed and how DRL can provide innovative solutions. This is

followed by challenges that researchers might encounter when applying DRL algorithms to UWS.

High trends of urbanisation globally increase the stress and demand on UWS with 60% of the world's population expected to live in urban areas by 2030 (UN-Water, 2012). This rise in demands causes heavier loads and more uncertainty throughout all processes in UWS due to increased supply and network expansions (Sharma *et al.*, 2010). Navigating these uncertainties can be challenging for meta-heuristic decision making algorithms (Maier *et al.*, 2014) in comparison to DRL algorithms that learn from experience and are able to act in real time (Fu *et al.*, 2022). DRL provides a method for managing uncertainties that outperforms traditional decision-making algorithms and can learn from experience which allows it to adapt to the rise in urbanisation.

Another challenge that plagues UWS is the energy consumption and carbon emissions associated with operating water systems (Nair *et al.*, 2014; Xu *et al.*, 2014). It was estimated that 1-18% of all energy consumed in urban areas is due to UWS (Olsson, 2012) which in return produces a lot of carbon emissions. The negative effects of high energy consumption lie beyond the financial impacts as it promotes climate change and global warming. The circular effect of carbon emissions, water scarcity and energy consumption is displayed in the water-energy-green house nexus (Nair *et al.*, 2014, fig. 1). DRL has had a proven record of improving energy management within the water systems (Hernández-Del-olmo *et al.*, 2016; Hernández-del-Olmo *et al.*, 2018) and in system efficiency (Kılış *et al.*, 2023).

UWS often deal with a heterogeneously aging infrastructure that add to the complexity of asset health management. The aging pipes, pumps, valves, and other system components can lead to high non-revenue water and effect the systems' overall resilience. Hence why, it is essential to provide decision making algorithms that can deal with high-level dependencies and complexities. A challenge that manifests with decision making algorithms is the high computational costs associated with this complexity thus why deploying DRL agents can benefit UWS as they rely on function approximators to lower the computational load (Sutton and Barto, 2018). Furthermore, asset management for UWS operations can be achieved by leveraging DRL for optimal design, strategic planning and predictive maintenance (Fu *et al.*, 2022). This area of research requires more experimentation and social proof despite its clear advantages.

In most pipeline infrastructure, it is necessary to quantify leakage and asset health. Managing leakage effectively is an ongoing battle that effects UWS especially water distribution systems.

The use of DRL for leakage management is an unrealised opportunity but has been recommended by reviews and surveys (Mosetlthe *et al.*, 2020; Fu *et al.*, 2022). The use of a tabular Q-learning method for leakage reduction using pressure management in water distribution networks was tested in (Negm, Ma and Aggidis, 2023b) and whilst the results were positive, it was clear that using DRL would enhance it further and overcome the curse of dimensionality.

3.3.2. Challenges of DRL in UWS

Building DRL algorithms is a science. In this section we build on the challenges and trade-offs underlined in the previous sections inherent in algorithm design. It is crucial to note that the field of RL research, much like the algorithms, has been expanded by experience followed by theory. In essence, some challenges were identified but not completely understood such as the deadly triad issue (Sutton and Barto, 2018).

In DRL algorithm design, most researchers will make use of some form of function approximation, bootstrapping or off-policy. Function approximation uses examples to generalise an entire function hence it aids with the scalability and generalisation issue that riddles tabular algorithms and is the main tide driving the success of deep neural networks in reinforcement learning (DRL). On the other hand, bootstrapping used in DP and TD fields help with improving the algorithm's data efficiency, hence reducing computational loads. Finally, off-policy methods free our agent from target policy to explore optimality. Separately, each of these methods help RL researchers reach their desired benefits and design a better optimisation algorithms, however when combined the same methods induce instability and divergence – the deadly triad issue (Tsitsiklis and Van Roy, 1997; Sutton and Barto, 2018). This instability can be detrimental when controlling urban water management system and could result in undesirable situation. Ensuring stability and resilience should be a primary goal of DRL design.

Another common challenge is the 'credit assignment problem'. This refers to the notable phenomena of incorrectly evaluating the credit of the action due to unclear or unforeseeable consequences manifesting later (Arulkumaran *et al.*, 2017). These long-term dependencies are necessary to allow the agent to better comprehend the value of its action. Hence, value functions have been modified to incorporate the estimated subsequent rewards and they have been discounted to signify the dwindling nature of consequence. UWS applications tend to be connected through both short-term and long-term dependencies therefore it is importance to include these consequences in the DRL algorithm's learning strategy.

Finally, the exploration versus exploitation dilemma. This problem riddles most RL (and DRL) algorithms as agents tend to behave in a reward greedy manner. Since the agent's observation depends on its actions and its actions depend on the reward generated; RL agents can find themselves in a loop around a local optimum rather than finding the global optima - exploitation. Ultimately, the only way to solve this is to introduce randomness to the agent's behaviour hence allowing the agent to receive new observations and possibly lead it to the global optima – exploration. This trade-off in agent behaviour has been navigated in many ways and the simplest is the use of ϵ -greedy exploration policy where the agent acts randomly with probability $\epsilon \in [0,1]$. The value of ϵ decreases as time passes leading the agent to a more exploitative nature as it learns. For continuous control, more complex methods have been used to introduce randomness over time to preserve momentum (Lillicrap *et al.*, 2016; Arulkumaran *et al.*, 2017). Other methods to tackle the exploration-exploitation dilemma include Osband *et al.*'s bootstrapped DQN using experience replay memory (Osband *et al.*, 2016), Usunier *et al.*'s exploration in policy space (Usunier *et al.*, 2017) and upper confidence bounds (UCB) (Lai and Robbins, 1985; Arulkumaran *et al.*, 2017; Pathak *et al.*, 2017). Managing the exploration-exploitation trade-off should be bespoke to each UWS application to ensure that agents don't converge at sub-optimal policies.

These challenges are inherent in most RL problems and navigating them is a skill necessary to develop an effective DRL algorithm.

3.4. DRL Research in Urban Water Systems

In essence, there are many parameters to consider when selecting a DRL algorithm but through careful consideration of selecting the correct DRL components and algorithms. Depending on the optimisation objective, the agent's nature (pump, valve) and requirements (nodal pressures, head measurements, pump speed) would vary. In a critical review of deep learning in the water industry Fu *et al.* mentioned the applicability of DRL in water distribution networks (WDN) and urban wastewater systems (Fu *et al.*, 2022). In (Croll *et al.*, 2023), the applications of reinforcement learning techniques in wastewater treatment were reviewed with a few studies utilising DRL methods. Otherwise, there are no mentions or reviews published on DRL algorithms in the water industry. There is limited literature on the application of DRL in UWS where most research relate to stormwater systems, water distribution networks and a few publications in wastewater systems. This shows a massive gap in the research field and an exciting journey for researchers in UWS at the cusp of realisation. In this section we will review the available literature on deep reinforcement learning in the water industry.

3.4.1. DRL in Water Distribution

In article (Hajgató, Paál and Gyires-Tóth, 2020), the authors use a Duelling Deep Q Network (D-DQN) to find the optimal pump speeds for hydraulic efficiency in randomly generated demands. The algorithm minimises the inflow and outflow of tanks whilst keeping heads within an acceptable range in all the nodes. The reward is calculated by evaluating the consumer satisfaction as the number of problematic nodes divided by the number of all nodes; the efficiency of the pumps as the product of standalone pumps divided by the product of theoretical peak efficiencies; the feed ratio by comparing the ratio of pumps supplying the water to the tanks and reservoirs supply. When compared to a test set of Nelder-Mead, Differential Evolution (DE), Particle Swarm Optimisation (PSO), Fixed-Step Size Random Search (FSSRS) and One-shot Random Trial; the agent performed at a comparable level to the differential evolution algorithm and much better than the rest of the test set. All the algorithms were tested on a small (Anytown) and large (D-town) WDN model. When using the one-shot random trial as a reference solution as a sub optimal policy; the agent reaches a better solution and moves off policy to overperform the DE algorithm. This technique relies entirely on live measurement data and can predict the best action in real-time making it the most suitable controller for real life application.

(Hu *et al.*, 2023) conducted a thorough experiment where they optimised the scheduling of fixed speed pumps to minimise the electric cost of the pumps and tank level variations whilst adhering to sensible hydraulic constraints using Proximal Policy Optimisation (PPO) and Exploration enhanced Proximal Policy Optimisation (E-PPO) (Hu *et al.*, 2023). Both DRL algorithms are policy-driven methods set out to find the best policy to achieve the highest rewards. They conducted three experiments that introduced three increasing levels of uncertainty to the consumer demand patterns using 0.3, 0.6 and 0.9 multiplier respectively on the Net3 test networks model. The results were compared with metaheuristics including genetic algorithms (GA), PSO and DE. GA converged after 100 epochs and were considered the optimal solutions (Hu *et al.*, 2023). They were followed in performance E-PPO followed by PPO, DE and PSO. The exploration enhanced policy saves approximately 6.10% of the energy cost with respect to PPO. Unlike the rest of the metaheuristic methods that require to be trained before each scheduling case; the DRL methods (PPO, E-PPO) can just call their trained models to act in a fraction of a second (0.4s) (Hu *et al.*, 2023).

(Xu *et al.*, 2021) tackles the pump scheduling optimisation problem in WDNs through combining knowledge learning and deep reinforcement learning in a knowledge assisted proximal policy optimisation learning (KA-PPO) (Xu *et al.*, 2021). KA-RL evaluates the state

using historical nodal pressure data and a reward function. Pressure management objectives were placed to maintain junction heads within a specific range, minimise water age, and increase pump efficiency. The proposed algorithm was tested on the benchmark Anytown network to manage the performance of two pumps in the pump station. The results show that the algorithm performs favourably in comparison to the Nelder-Mead method and the DDQN algorithm used in (Hajgató, Paál and Gyires-Tóth, 2020; Xu *et al.*, 2021). Future work can improve the reward formulation process by including energy prices. The problem setup can also be modified to consider a continuous action space and long period accumulated return. The use of emulators and parallel computing can also minimise the training time.

In (Hasan *et al.*, 2019), the authors offer four novel contributions to the fields of dynamic multiple-objective deep reinforcement learning and water quality resilience applications. Based on the deep-sea treasure (DST) test bed, the authors develop a new test bed to fit the RL settings hence creating the first test bed accommodating for dynamic multi-objective DRL (DMODRL). They also devise a new for multi-objective optimisation using DRL and the first deployment of objective relation mapping (ORM) to construct the govern policy (Hasan *et al.*, 2019). The last contribution is an expert system to evaluate the water quality resilience (WQR) in Sao Paulo, Brazil. The proposed parity-Q deep Q network (PQDQN) algorithm proposed was tested in the two DST environments and the WQR model. In all three test beds, the PQDQN algorithm has outperformed the state-of-the-art multi-policy DRL algorithms which were multi-policy DQN (MP-DQN), multi-objective monte carlo tree search (MO-MCTS) and multi-pareto Q learning (MPQ). In all three test beds, the performance of the algorithms were assessed using the evaluation matrices generational distance measure (GD), inverted generational distance (IGD) and hypervolume (HV) (Hasan *et al.*, 2019). PQDQN managed priorities best using the ORM aiding its impressive performance and defeating the other multi-policy algorithms (MP-DQN, MO-MCTS, MPQ) (Hasan *et al.*, 2019). This work can benefit by experimenting with multi-agent DRL and integrating real-world scenarios to the WQR model. Parallel computing and GPU processors can also reduce training time. Hyperparameter optimisation may even improve the performance of the PQDQN algorithm further.

In a broader look on water systems, (Fan, Zhang and Yu, 2022) tackles asset management of water distribution networks post-earthquake. The problem setup involves four models that assess damages incurred by the earthquake, recover the water distribution network (WDN) using the optimisation algorithms, measure the WDN hydraulic performance using the performance degree (PDW) at each timestep, quantify the overall WDN resilience using the system resilience index (SRI). The chronological and iterative process between these models is

clearly displayed in (Fan, Zhang and Yu, 2022, fig. 2). A graph convolutional network (GCN) was deployed as the function approximator for a DQN algorithm hence creating GCN-DQN. This selection was a great step towards better representation for water distribution networks since the graphical nature of the data requires a similar deep neural network architecture. Other strategies used for comparison included two greed search algorithms (static importance based and dynamic importance based), genetic algorithm (GA) and diameter-based prioritisation method. All five strategies were tested under three identical earthquake scenarios with different magnitudes. In all three scenarios the GCN-DRL model outperforms the other strategies by following repairing sequences that lead to higher SRI scores (Fan, Zhang and Yu, 2022). The importance-based methods came second and third whilst the diameter-based prioritisation came last. In order to minimise the training computation time, the authors have used transfer learning to use the previous GCN weights on an old damage scenario to initialise the GCN weights for the new scenario. This reduced the computational load significantly and proved the scalability of the GCN-DRL model across all scenarios. Accommodating more sophisticated assumptions can be easily implemented to improve the GCN-DQN model's reliability and improve the problem setup. Applying this work on different test networks can further prove its generality and encourage more development of asset management through deep reinforcement learning.

3.4.2. DRL in Stormwater Systems

Mullapudi et al. provide a first look on the application of deep reinforcement learning for real time control in storm water systems (Mullapudi *et al.*, 2020). The authors test a simple DQN algorithm on the urban watershed in Ann Arbor as a benchmark test network. The problem setup involved agents taking actions to control valves status; water levels and outflows as states and an assumption of uniform rainfall and negligible base flow (Mullapudi *et al.*, 2020). The authors set out to test the stability of DRL algorithms in controlling storm water management models (SWMM) through controlling a singular basin and controlling multiple basins. Their research highlighted DRL algorithms' known sensitivity to reward formulation and deep neural network architecture. Even though the agent could have benefitted from a longer learning phase, the DRL proved useful in managing the single-basin SWMM scenario. Due to the increase in state and action space, controlling multiple basins was more challenging. The agent behaved favourably in comparison to uncontrolled SWMMs in both scenarios but were outperformed by the equal-filling algorithm. The authors remain determined that RL-based controllers need to be explored further and applied to SWMM in hopes of reaching a stable real-time controller. The results provided in this paper could be used as a starting point to

compare more capable DRL algorithms A3C and advanced variations of DQN. Also, a more systematic method for reward formulation and neural network hyperparameter optimisation would greatly improve the scalability and stability of the model.

A common issue with real-time control using DRL is concerns of the reliability and uncertainty of its fluctuating actions in high-risk real-world cases. Tian et al.'s paper tackles this issue through a novel methodology called 'voting' (Tian, Liao, Zhi, *et al.*, 2022). Voting compares actions from five different DRL algorithms to select the safest and most rewardable action hence minimising the risk associated with DRL control. If none of the DRL agents provide a viable action, a backup user-defined rule-based action is executed. The methodology is used to minimise combined sewer overflow (CSO) and flooding in urban drainage system. The DRL algorithms used in this study are DQN, DDQN, PPO1, PPO2 and A2C. Voting uses a novel independent security system to evaluate whether the actions meet the user-defined safety requirements. All five DRL algorithms and voting algorithms are compared to a GA algorithm that was used as an upper bound performance reference by subjecting them to eight scenarios under different rainfall patterns. The results prove that voting avoids harmful actions to minimise risk hence improving the reliability of the real-time control. Figure 16 highlights that voting often draws its actions from PPO1 and never needed to use the backup action in all eight scenarios (Tian, Liao, Zhi, *et al.*, 2022, fig. 16). All DRL algorithms have performed well in this sequential problem and are therefore suitable candidates for CSO and flooding mitigation. Concerns of long training times and computational loads can be mitigated with parallel computing and an emulator for the stormwater model. The DRL algorithms can benefit from hyperparameter optimisation to improve the results further. Future work can also attempt deploying the voting algorithm on a SCADA system or online monitoring system to uncover uncertainties from real world applications.

It is worth mentioning that the authors published a different paper where they developed an emulator for the stormwater model to relieve the high computational load associated with training the DRL agents (Tian, Liao, Zhang, *et al.*, 2022). This emulator succeeded in decreasing the training time by 9 hours and 57 minutes hence improving data efficiency when compared to the regular RL-stormwater model approach.

Like the previous article, (Bowes *et al.*, 2021) leverages the power of DRL for flood mitigation. In this experiment, the authors developed a DDPG algorithm to create control policies that mitigate flood risks in the coastal city of Norfolk, Virginia. The DRL agent manages to balance flooding throughout the system and follow the control objectives of maintaining target pond

levels and mitigating flood through controlling valves in the stormwater management model. The performance of DDPG as a DRL method was compared to rule-based control strategy, model predictive control and a passive system. In summary, the DDPG algorithm boasted a 32% reduction in flooding in comparison to the passive system and a 19% reduction with respect to rule-based control. The model predictive control strategy deployed an online genetic algorithm optimisation as in (Sadler *et al.*, 2020) to produce similar results to the DDPG algorithms (3% reduction in flood compared to DDPG). The model predictive control was too computationally expensive to run on the complete dataset whilst RL provided an 88x speed up in the creation of control policy (Bowes *et al.*, 2021). This research highlights the power of DRL in real-time control of stormwater systems and its ability to produce impressive results with a lower computational load. Further research should aim to recreate these results on real-world systems through RL controllers. Combining the different real-time control methods as decision support tools should be investigated to enhance stormwater systems.

3.4.3. DRL in Wastewater Treatment

Wastewater treatment has initially experimented with RL methods to manage the oxidation-reduction potential and pH levels of wastewater using Model Free Linear Control (MFLC-MSA) (Syafiie *et al.*, 2011), improve the cost of N-ammonia removal using tabular Q-learning (Hernández-Del-olmo *et al.*, 2016), improving energy and environmental efficiency of N-ammonia removal using policy iteration (Hernández-del-Olmo *et al.*, 2018), and optimising hydraulic retention through aerobic and anaerobic processes for biological phosphorous removal using Q-learning (Pang *et al.*, 2019). In addition, actor critic RL methods are utilised for pH adjustment for electroplating industry wastewater in a continuous action space (Alves Goulart and Dutra Pereira, 2020). This RL method was mimicked in (Yang *et al.*, 2022) where the authors utilise an actor critic RL method to track the desired dissolved oxygen set points in a wastewater treatment plant (WWTP). A more detailed review of RL application in WWTP can be found at (Croll *et al.*, 2023). Following the successes of DRL algorithms and its growing popularity, more research has deployed DRL methods to solve issues in WWTPs.

The only use of value-based DRL algorithm in wastewater treatment is present in (Nam *et al.*, 2020). The article carries out an experiment involving both RL (Q, SARSA) algorithms and DRL (DQN, deep-SARSA) to reduce the aeration energy consumption without decreasing the effluent quality index. These factors were estimated using the activated sludge model soluble product (ASM-SMP) named benchmark simulation model 1 (BSM 1) developed by (Alex *et al.*, 2018). The DQN model largely outperformed the other methods as it develops a trajectory that simultaneously improves the economic benefits by 36.53% and the environmental

efficiency by 0.23%. The RL methods deployed fail to handle the complexity and caused decreases in energy savings and environmental efficiency. Further work recommended includes the experimentation with multi-agent systems to control environmental and economic benefits whilst minimising risks from membrane fouling (Nam *et al.*, 2020). The authors did not discuss hyperparameter optimisation which could further improve their current results. In addition, the use of policy gradient methods can provide insights on the difference in policy gradient and value driven DRL in performance.

In (Panjapornpon *et al.*, 2022), the author leverage the hybrid properties of multiple DDPG agents as an actor critic method. This study is more focused on developing a MADRL for pH control and tank level control by simultaneously managing the flow rates of the influent stream and neutralisation stream (Panjapornpon *et al.*, 2022) in a continuous stirred tank reactor. The authors use the grid search methods for hyperparameter tuning of three performance indexes. The DDPG uses a gated recurrent unit and rectified linear units for the actor and critic networks as shown in figures 6 & 7 (Panjapornpon *et al.*, 2022, figs 6 & 7). The multi agent DDPG algorithm performed favourably in comparison to the proportional-integral controller with controlling efficiency with better performance indexes and less oscillations (Panjapornpon *et al.*, 2022). This paper highlights the benefits of using DRL to optimise control performance. Deploying the RL controllers using programmable logical controllers on real WWTPs can provide social proof.

MADRL is utilised in (Chen *et al.*, 2021) to control dissolved oxygen set points and chemical dosage in WWTP. In this article, the authors use a multiple agent DDPG algorithm to lower environmental impacts, cost and energy consumption using a life cycle driven reward function. The life cycle assessment driven strategy has outperformed cost oriented and effluent quality optimisation in eliminating environment impacts. The use of multiple agent DDPG has provided good results however the study lacks comparisons with other optimisation algorithms which should be investigated in the future. MADRL should enable better navigation in highly complex environments therefore it would be great to validate this novel algorithm with field data.

A statistical learning based PPO algorithm is used to develop a predictive control strategy that minimises energy consumption in a wastewater pumping station in (Filipe *et al.*, 2019). The model free method decreases electrical consumption by 16.7% and tank level violations by 97% in comparison to the current operating conditions of the pumping station based in a WWTP in Fábrica da Água de Alcântara, Portugal. The authors also compare the results of

using wastewater intake rate forecasts to improve the PPO algorithm's results. Indeed the forecasts help improve the results of the algorithm with cumulative energy consumption dropping from 459MWh-469MWh to 340MWh-348MWh (Filipe *et al.*, 2019). Bayesian optimisation was also utilised to optimise the forecasting hyperparameters. It is important to compare these results to other model predictive control methods used in WWTP pumping stations and other optimisation approaches to highlight the DRL algorithm's performance with respect to known benchmarks. It will be beneficial to recreate the results using WWTP benchmark models and validate the results in real-world applications.

3.4.4. DRL in Raw Water Treatment

There haven't been many papers to review relating to the application of DRL to the supply and treatment of raw water. A related paper discusses the use of DRL as a smart planning agent for off-grid camp water infrastructure (Makropoulos and Bouziotas, 2023) therefore it is not an urban water system. DQN, PPO and multi-armed bandits were tested using an urban water optioneering tool (UWOT). The DRL agents are tasked with using an array of different supply technologies with relevant costs and a set of demand pattern for potable and non-potable water to explore conditions of deployment in the off-grid system. This paper's ability to train and test DRL agents in strategic planning paves the way for strategic planning opportunities in UWS as well.

The only raw water supply application can be found in (Z. Li *et al.*, 2023) where the researchers apply proximal policy optimisation (PPO) algorithm to lower suspended sediment concentration (SSC) and energy consumption tested on data from the Yellow River pumping station in China. The DRL environment is made by combining data from the hydraulic model and the SSC predictive model which is formed of a multilayer perceptron model. The PPO algorithm is trained on the predicted SSC (predictive control) and real-world SSC data (perfect predictive control). Both strategies are compared to manual strategy developed by experienced operators. The SSC predictive model was not accurate as it deviates from the training and validation sets. In both the predictive and perfect predictive control, the DRL algorithm outperforms the manual strategy resulting in a smoother sediment profile, decreases the energy consumption by 8.33%, and average sand volume per unit water withdrawal by 37.01% and 40.575% respectively (Mullapudi *et al.*, 2020). Furthermore, the authors investigate the effects of reservoir water outflows and initial reservoir water volumes. There is a strong relationship between reservoir initial water volume. This paper can benefit by comparing the DRL algorithm to other heuristic optimisation algorithms such as iterations of genetic algorithm (GA) or differential evolution (DE). The researchers should attempt to

optimise the reward function by experimenting with different weights and apply some form of hyperparameter optimisation to increase the accuracy of the SSC predictive model.

Table 3-1 Summary of reviewed articles

System	Application	Algorithms	Case Study	Remarks	Reference
Water Distribution	Pump control	DDQN	D-town, Anytown	DDQN controls pump speeds to minimise tank outflows and keep junction heads within an acceptable range.	(Hajgató, Paál and Gyires-Tóth, 2020)
		PPO, E-PPO	EPANET Net3	E-PPO achieves the better performance in minimising tank level fluctuations and pump energy consumption.	(Hu <i>et al.</i> , 2023)
		KA-PPO	Anytown	KA-PPO controls pump speed to keep junction heads in acceptable range, minimise water age and increase pump efficiency	(Xu <i>et al.</i> , 2021)
	Water quality	PQDQN	Sao Paolo, Brazil	A novel DST and WQR expert system for DMODRL. PPQN outperforms the other algorithms.	(Hasan <i>et al.</i> , 2019)
	Asset management	GCN-DQN	Rancho Solano Zone III	Novel problem setup to test resilience post-earthquake. Use of GCN as function approximator and transfer learning greatly improves results.	(Fan, Zhang and Yu, 2022)
Stormwater systems	Flood control	DQN, DDQN, PPO1, PPO2, A2C, Voting	Sewer system in eastern China	Novel method to improve the reliability of DRL algorithms (voting). Novel emulator that outperforms benchmarks in modelling storm water systems.	(Tian, Liao, Zhi, <i>et al.</i> , 2022)
		DDPG	Norfolk, Virginia, USA	DDPG used for flood mitigation in real-time. Better results than rule-based control and faster than model predictive control by 88x.	(Bowes <i>et al.</i> , 2021)
	Valve control	DQN	Ann Arbor	DQN algorithm successfully controls SWMM but raises issues of reliability for real-world application. Serves as a starting point for further research.	(Mullapudi <i>et al.</i> , 2020)
Wastewater systems	Dissolved oxygen settings	Deep SARSA, DQN	BSM 1	DQN algorithm outperforms all RL and DRL methods used to simultaneously increase environmental efficiency and minimise energy consumption.	(Nam <i>et al.</i> , 2020)
		Multi agent DDPG	Jiangsu Province, China	Life cycle assessment proven as a superior reward function for a multi agent DDPG in minimising environmental impact.	(Chen <i>et al.</i> , 2021)
	Pump control	PPO	Fábrica da Água de Alcântara, Portugal	WWTP pump control using wastewater intake rate forecasting to improve energy efficient and tank level violations with respect to normal operating conditions.	(Filipe <i>et al.</i> , 2019)

	pH control, tank level control	Multi agent DDPG	Servo-regulatory MATLAB test	Multi agent DDPG used to improve real time control of pH and tank levels with respect to a proportional integral controller.	(Panjapornpon <i>et al.</i> , 2022)
Raw water supply	Sediment control	PPO	Yellow river pumping station	PPO outperforms experts' manual strategy and decreases energy consumption by 8.33%. Should be compared to other optimisation algorithms.	(Z. Li <i>et al.</i> , 2023)

3.5. Future Work and Novelties

As repeatedly displayed throughout this review, the field of deep reinforcement learning is growing rapidly and expanding across various real-world applications; the most recent of which being the water industry. This field of application is relatively new and is brimming with new possibilities for the real-time control. Extending this technology to the operational management of water systems is a field of untapped potential with many avenues to explore. DRL provides a method to continuously train the model to react and adjust to the environment it is placed in. This ability for unsupervised learning makes DRL a great tool for the instantaneous optimisation of any foreign network hence possibly globalising it water networks across the country. Researchers are therefore encouraged to experiment with simple DRL algorithms in different aspects of water distribution networks, stormwater systems, water treatment and sanitation, wastewater management such as strategic planning and asset management. The link between leakage and greenhouse gas emissions has been repeatedly mentioned in water management literature (Negm, Ma and Aggidis, 2023a) due to its relevance in the research community. It will be interesting to extend DRL algorithms in water applications to minimize carbon emissions.

As this is the first review dedicated to deep reinforcement learning in UWS, the collation of this evolving field should be constant to act as a beacon to new researchers. More review papers will also help define the community's direction, evaluate recent findings, and reveal possible novelties. Nevertheless, it is essential that researchers interested in this field spend a considerable amount of effort understanding the fundamentals of DRL. This will help clear any misconception on the applicability of the field and highlight any new advancements. Hopefully, this will steer academics away from repeating mistakes. More research articles with the purpose of formalising methods of DRL application would serve as a great bridge for aspiring researchers. Whilst researcher focus on testing DRL on models and software case studies, it is necessary to validate the use of DRL as controllers in real-world case studies. Finally, focusing on the application of DRL in graphical based distribution systems such as the electrical distribution networks will provide a clearer perspective on possible overlaps and trends that could benefit water distribution.

To fuel further research, the research community should focus its efforts on benchmarking scalable DRL environments for testing. Early efforts to benchmark environments can save upcoming researchers the need to repeatedly contextualise the optimisation problem in the scope of DRL. These environments should be able to communicate effectively with the most popular hydraulic simulators (e.g., EPANET, SWMM and so on) through wrappers such as

PYSWMM (McDonnell *et al.*, 2020) and EPYNET (Vitens, 2017). They should also be written in the necessary syntax to include benchmarked DRL libraries such as Stable Baselines, PyTorch, TensorFlow and so on. As this is an engineering application, researchers should aim to develop models that focus on reliability and scalability. Demonstrations of these algorithms acting on live data and ground-truth models in real-time should be the objective from an engineering perspective.

3.6. Concluding Remarks

After introducing the proposed field of DRL in the water industry, the field was contextualised in the realm of artificial intelligence and machine learning. The main advantages and properties of reinforcement learning were highlighted to explain the appeal behind the technology. This was followed with a gradual explanation of the formalism and mechanisms behind reinforcement learning and deep reinforcement learning supported with mathematical proof. Different computing fields were explained thoroughly to highlight the origins of commonly used computing methods in DRL. Furthermore, the milestones, trends and challenges of deep reinforcement learning were discussed to develop a better understanding of the current research area. The main research articles that have adapted deep reinforcement learning methods to solve problems in urban water systems were reviewed thoroughly and summarised in **Table 3-1**. Finally, future works and recommendations were included to provide a clear view for the application of DRL in the water industry. The main conclusions from this section can be described as follows.

Deep reinforcement learning improves on reinforcement learning using deep neural networks for function approximation. This has improved scalability and resulted in many successes across simulated and real applications.

Current DRL trends tackle high dimensional complexity by mimicking human psychology and natural hierarchy structures.

The field of deep reinforcement learning can benefit from better classification to help new researchers navigate better.

The application of DRL in the water industry is still in its infancy yet it shows great promise to improve our current practices with water. Early efforts to benchmark DRL test beds and environments will aid the growth of this topic.

The use of DRL as a method for pressure management in water distribution networks has not been tested before yet it shows promise due to DRL's ability to tackle dynamic scenarios in

other UWS applications. We believe that applying DRL as pressure management strategy could bring us closer to real-time leakage prevention and increase the resilience of our WDNs.

4. Water Network – Deep Reinforcement Learning Ecosystem

The water distribution network – deep reinforcement learning (WDN-DRL) ecosystem is the name given to our data architecture used to simulate real and modelled leakage scenarios through a closed-loop system between the hydraulic solver and the optimisation algorithms. Establishing the tools necessary to set this architecture and defining the parameters of the leakage scenarios were essential predecessors to developing an effective ecosystem. Hence why, the research discusses the design and build choices necessary for the development of this WDN-DRL ecosystem.

4.1. The Leakage Problem

The leakage problem is a well-researched issue that plagues most water distribution networks and appears in two general forms. Background leakage is the term given to small leak events that lie under the level of detection using our current technology. Their undetected nature makes them dangerous as they cannot be mitigated using active leakage control (ALC) strategies and must solely on pressure management. Despite their low magnitude, multiple occurrences of background leak events are registered through effective leakage assessments and their resulting increase in non-revenue water. They can be further proven through discrepancies between accurate network models and real models.

Alternatively, burst events are a result of a noticeable failure in WDN infrastructure that is often easily detectable due to its repercussions on the hydraulic properties of the network. Hence why, burst leakage can benefit from both ALC and pressure management. These events, despite their severity, are often handled quickly through a dispatch of technical help and advanced asset management to decide whether replacing or repairing the failed infrastructure is the favourable option for leakage control. Therefore, pressure management of such events serve as a temporary yet effective contingency until the burst location is identified and resolved. This process of detecting, locating, and resolving burst events can vary from hours to months depending on the utility's strategy which further proves the benefits of introducing pressure management as a contingency to minimise the financial and environmental burden of burst events. Advanced pressure management is achieved by ensuring two main objectives which are minimising leakage rates across all nodes and ensuring nodal pressure remain between 10m to 70m to avoid OFWAT (regulatory body) DG2 violations (OFWAT, 2004). Nodal pressure limits help maintain water supply to consumers without affecting asset life in accordance with the regulatory body OFWAT.

4.1.1. The Hydraulic Model

Model Building

Understanding leakage can only be achieved by observing the hydraulic effects of those events throughout the WDN through hydraulic analysis. Hydraulic models of WDN are often created to achieve that using software derived from electric grid systems (Walski, 2003). These models are assembled by layering geographical data such as elevation and network maps, intrinsic network data such as pipe information (diameter, length, roughness) and network components, operational data such as pump speeds and valve settings, customer data such as demand patterns to simulate the hydraulic properties of the WDN. This information was initially sourced from system maps however Geographical Information System (GIS) have been the most popular choice in recent years. GIS model links thematic layers of data together geographically to determine relationships between data. Therefore, it can integrate query and statistical analysis with geographical benefits offered by maps (ESRI, 2023). Operational data can be extracted from SCADA (Supervisory Control and Data Acquisition) systems from water utility control systems. Finally, data is also collected from Customer Information Systems (CIS) to measure or predict customer demand pattern. Data was gathered from test networks in the research community and Northumbrian Water utility to develop the hydraulic models for the experiments.

Hydraulic software tools are readily available among water utilities and the research community, with options available for both free and commercial versions. Some hydraulic solvers can accommodate for different fluids with different hydraulic properties and multi-phase flow simulations. However, this research is only concerned with modelling water (single-phase) flow. In addition, several of the commercially available software include an in-built leak detection module that uses optimisation strategies to identify leaks. A summary of hydraulic model platforms for pressurised fluid pipelines and pipe networks can be found below in **Table 4-1**.

Table 4-1 Hydraulic Solvers

Hydraulic software	Fluid type	Leak detection module	License type
Single – phase flow			
EPANET	Water	Unavailable	Free
LOOP	Water	Unavailable	Free
CADRE flow	Water	Unavailable	Commercial
Pipe Flow Expert	Water	Unavailable	Commercial
Synergi Pipeline Simulator	All fluids	Available	Commercial
InfoWorks WS	Water	Available	Commercial
WaterGEMS	Water	Available	Commercial

For the purpose of this research, it was necessary to use the benchmarked hydraulic solver - EPANET- as it is widely used in both research and industry making it a suitable option. In addition, the availability of application programming interfaces (APIs) between EPANET and python aids in the communication between the bespoke environment and the EPANET software.

Introducing Leak Events

As mentioned in leakage management literature, leakage events are often described in terms of burst events and background leakage where background leakage is the name coined for leakage events lie under the detectable range. In the UK, this is assumed to be leakage with flow rates less than 0.5m³/h (equates to 1.39 L/s) in the pressure of 50m and a leakage exponent of 0.5 (García and Cabrera, 2007). EPANET computes friction headloss using the Hazen-Williams formulas. It also uses the pressure dependent leakage equation (2-9) rewritten below, rather than the FAVAD equation to compute leakage flow rates. Essentially, EPANET introduces leaks by changing the emitter coefficient.

$$Q_l = kP^n \quad (2-9)$$

Where Q_l is the leakage flow rate (L/s), P is the nodal pressure (m), k is the emitter coefficient ($Ls^{-1}m^{-0.5}$), and n is the leakage exponent. Rearranging, **equation 2-9** can help us determine the threshold for emitter coefficient that model background leakage (**Eq. 4-1**).

$$k = Q_l/P^n \quad (4-1)$$

Using the figures mentioned in (García and Cabrera, 2007), we can conclude that background leakage can be denoted as leaks with emitter coefficients less than 0.196 and burst events

have emitter coefficients greater than 0.196. This coefficient marks the boundary between detectable leakage (bursts) and undetectable (background) as calculated from (García and Cabrera, 2007) . The emitter coefficient on each node is also calculated using equation (3-2) below.

$$k = c \times \sum_{j=1}^M 0.5 \times L_{ij} \quad (4-2)$$

Where k is the emitter coefficient ($Ls^{-1}m^{-0.5}$), c is the discharge coefficient ($Ls^{-1}m^{-1.5}$), M is the total number of nodes in the network, and L_{ij} is the length between the nodes i and j (m). The discharge coefficient depends on the shape and the diameter of the neighbouring pipes.

Another important decision was the pressure exponent (n) which varies between 0.47 to 2.5 depending on soil properties, pipeline characteristics and failure type (Walski *et al.*, 2009). The original use of the equation is to estimate the laminar flow through an orifice which uses a leakage exponent of 0.5 or transitional flow with a leakage exponent between 0.5 and 1. However, laminar, and transitional flowrate must be very small and insignificant which does not accurately reflect practical WDN operations. Studies have proven that the range in leakage exponent can be explained with the pressure-induced variable leakage area (Darweesh, 2022).

Hydraulics Options		Junction 2	
Property	Value	Property	Value
Flow Units	LPS	*Junction ID	2
Headloss Formula	H-W	X-Coordinate	-954.479
Specific Gravity	1	Y-Coordinate	8017.621
Relative Viscosity	1	Description	
Maximum Trials	40	Tag	
Accuracy	0.001	*Elevation	18
If Unbalanced	Continue	Base Demand	10
Default Pattern	1	Demand Pattern	Fc
Demand Multiplier	1.0	Demand Category	1
Emitter Exponent	1.18	Emitter Coeff.	0.033656
Status Report	Full	Initial Quality	
Max. Head Error	0	Source Quality	

Figure 4-1 Modifying emitter coefficient and leakage exponent in EPANET.

Consequently, in our experiments we adopt the assumption that $n=1.18$ which is widely-used in leakage control literature (Araujo *et al.*, 2006; Saldarriaga and Salcedo, 2015b). The emitter exponent describes the overall state of the network where higher exponents refer to aging and less resilient assets. Therefore, a higher emitter exponent would indicate higher leakage rates as seen in **equation 2-9** and vice versa.

Figure 4-1 demonstrates how these modifications in emitter coefficient and leakage exponent appear in the Epanet software. For the burst scenarios, an emitter coefficient between $1 Ls^{-1}m^{-$

0.5 and $10 \text{ L s}^{-1} \text{ m}^{-0.5}$ will be selected for three randomised nodes to model the effects of failure events caused by large bursts whilst background leakage will be modelled using a range of discharge coefficients between $0 \text{ L s}^{-1} \text{ m}^{-0.5}$ and $0.194 \text{ L s}^{-1} \text{ m}^{-0.5}$ for all the nodes across the network to emulate a WDN with undetected background leakage.

4.1.2. Markov Decision Process and RL Context

Before designing the WDN-DRL ecosystem, the leakage problem must be redefined using the Markov decision process (MDP) formalism. Additional definitions of necessary RL terms will be contextualised for the leakage problem. This section will prove as a useful preamble to creating the central python environment.

Markov decision processes (MDPs) is a mathematical framework that effectively models stochastic decision making in dynamic scenarios where the result is determined through sequential actions. They are the foundational theory behind reinforcement learning (and deep reinforcement learning). The notations used in MDP include reward, agents, actions, state, and environments which serve as a basis for the second-order notations of RL terms such as observation, episode, and policy. MDP is formed through iterations of the original Markov Process (MP). After introducing rewards to make Markov Reward Process (MRP), actions are incorporated to form the Markov Decision Property (MDP). The general Markov property stipulates that the next state can be determined solely from the current state, which holds all the relevant information of the past (Lapan, 2019). MDP includes five main components denoted by the five-tuple (S, A, P, R, γ) that stand for states (S), actions (A), probabilities/dynamics (P), reward (R) and initial state (γ) (Puterman, 1990; Desharnais *et al.*, 2004) and can be explained through the graphical representation in **figure 4-2** below.

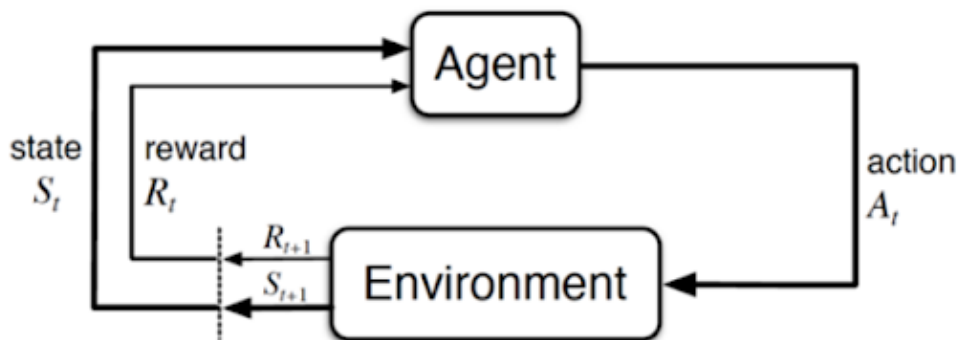


Figure 4-2 Fundamental MDP model

The agent is the physical component that interacts with the environment through actions, taking into consideration the environmental states and receiving rewards. In engineering applications, this tends to be an actuator. In our experiments, the agent is a garrison of

pressure reducing valves (PRVs) optimally placed in the network and/or variable speed pumps (VSPs).

The action (A_t) denotes the steps the agent can take to influence the environment, altering the environment's state and the perceived reward. These actions are often modelled after the actuator's possible movement. In our experiments, each action simultaneously sends signals to control distinct PRV settings or both PRV settings and VSP speeds. This simultaneous control provides the ability to replace the function of water utilities' control rooms and enables our agent to take full control of the network during each iteration.

In the most general sense, the environment is the everything that exists outside the agent (i.e., the universe) however we limit the environment to the surroundings that interact with the agent. Our scope limits the environment to the water distribution network it is placed in. The WDN is modelled in EPANET to include all the necessary data from practical scenarios to effectively train and test the agent. The environment provides the agent with dynamic transition probabilities that account for the three-dimensions of initial state, target state and action. Using the provided action, the environment can compute the next state using its transition probabilities to reach a new state and the resulting reward. Due to the multiple functions in the environment, it is regarded as the centrepiece of DRL application.

The state (S) is an overarching representation of the current environment. It must include the relevant features of the environment for the agent to make a decision. Water distribution networks share two main features which are the average pressure and flow in the system. In the context of our problem, a third figure for leakage is necessary. After each action, the state is updated using the environment's transition probabilities and a new state is sent to the agent.

Rewards are either dependent on the new state produced, or more commonly, on the state-action pair. The purpose of this numerical value is to grade the agent's behaviour, providing critical feedback to improve our agent. In RL theory, this reward is used to calculate the mathematical expectation of return, forming state value and state-action value which is explained in detail in section 3.1. Reward formulation is a crucial part of the optimisation process and requires multiple iterations to perfect. It should model our desired outcome perfectly and include concessions for unexpected behaviour such as the exploration-exploitation dilemma. Hence, we calculate our reward through minimising pressure violations depicted in OFWAT's DG2 (OFWAT, 2004) and minimising leakage rates. Our main objectives are to keep nodal pressure above 10m as recommended by DG2 violations (OFWAT, 2004) and

minimise leakage rates. A discount factor is also used to tune the agent's foresight and avoid exploitation of the short-term reward. This prevents convergence at local minima or excessive exploration which could lead to non-convergence.

Other DRL terms are commonly used to explain the problem setup which are useful to redefine for the application in WDN. For instance, policy is the learnt probability distribution of actions across each distinct state. It commands the agent's behaviour by suggesting the action to each state it encounters. Achieving the target policy is the primary goal of each DRL algorithm as it ensures that the agent will always act in a desirable and optimised manner. Consequently, a lot of emphasis has been placed on bridging the gap between the behavioural policy and target policy through policy-driven algorithms. Some examples include proximal policy optimisation and hybrid policy as well as value driven algorithms such as actor-critic algorithms (see **Figure 3-2**). It is essential to note that this scenario requires the use of stochastic policy, which, unlike deterministic policy, chooses the following action from a probability distribution of the action space rather than a single unwavering action. This allows the agent to account for the uncertainties arising from the knock-on effect of leakage events in WDN operations, rather than assuming a deterministic path. The target policy selected for this problem setup is advanced pressure management for leakage and violation minimisation.

Observation is a DRL term that overlaps mostly with the definition of state in MDPs. It is the relevant information received by the agent to aid with forming the policy and selecting the consequent action. The observation influences the agent in a similar manner to the state in MDPs. In RL language, state is used to describe the current iteration of the environment and help communicate with the user whilst observations are used to guide the agent's learning. The observations used for the pressure management in WDN include the changes in leakage rates after introducing new PRV settings and the full list of PRV and VSP settings. This will provide the agent with a distribution of leakage rates to be solved and its current settings.

DRL problems are often described as episodic or non-episodic which describes the sequential behaviour and terminating conditions of the problem. Non-episodic problems continue to run until the agent reaches a terminating state (e.g., losing or winning a continuous game) whilst episodic problems run until a time limit or iteration limit has been reached- completing the episode. In the interest of managing WDNs, we treat the problem in an episodic manner that models a day with new setting introduced each hour thus forming 24 distinct timesteps. This matches the operations of an advanced WDN control room as most water utilities only modify PRV settings daily.

Each paragraph in this section, redefines the relevant MDP and RL terms to provide context to the pressure management problem. This supports the formation of the schematic used to design and build the WDN-DRL ecosystem shown in **figure 4-3** below.

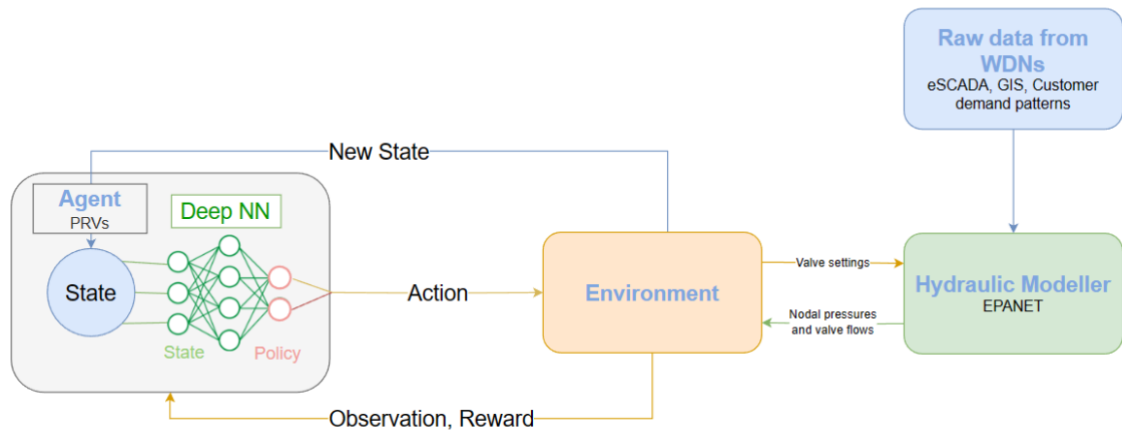


Figure 4-3 WDN-DRL ecosystem schematic

As mentioned, the ecosystem starts by building a justified hydraulic model of real-life networks through data provided from Northumbrian Water or research benchmark test networks to provide the necessary transitions through EPANET which acts as a hydraulic solver software. Our contribution is the environment and agent. The environment will be created in the next section by extracting the relevant data from the hydraulic model and processing it in terms of RL formalism such as state, observation, and reward. The environment will also relay the actions provided by the agent’s algorithm in terms of processed valve settings or valve and pump settings. The agent will be optimised using different neural network architectures to test varying DRL algorithms. In our research, the design and use of the agents and environment are the main contributions to the wider research community.

4.2. The Environment

An environment is the world that the agent interacts with. It provides the context and the dynamics in which the agent operates, and it plays a crucial role in the learning process. The environment also defines the state space, action space, and the rules for transitioning between states and receiving rewards. Designing a decent reinforcement learning environment must effectively challenge and test the learning capabilities of the RL agents while providing a meaningful and well-defined task. Therefore, the environment must involve several important considerations.

The environment must have clear objectives that are specific, measurable, and relevant to the leakage problem at hand. The objectives must reflect the desired aims of the leakage problem.

This is achieved through a well-designed reward function that aligns with those aims and encourages the desired behaviour. The reward function must provide insightful feedback to the agent. The environment should be challenging enough to require learning and adaptation but not so difficult that the agent cannot make any progress. It must have observable states that provide sufficient information for the agents to make an informed decision. As such, it is important to choose an effective observation space. Similarly, the action space design must be appropriate for the task. It should allow the agent to express a variety of behaviours and strategies.

In addition, limitations must be placed to avoid catastrophic failures during training. This is especially important in real-world applications like water distribution. Other considerations for real-world applications include realism where the environment should be representative of the problem that the RL is intended to solve. Hence, emphasis was placed on exploiting the hydraulic analysis capabilities of software such as EPANET. It is essential that the environment is reproducible and scalable so that experiments are fair and feasible to train agents using available resources. Creating or selecting the right environment for your RL task is a critical step in the RL pipeline. The reinforcement learning community has developed benchmarks for custom environments using the OpenAI Gym framework and DRL agents through Keras RL and Stable baseline libraries. Using the benchmarks should make the environment accessible and available to researchers and developers.

A special consideration in designing the environment was to ensure that other optimisation algorithms can be run using the same environment. The environment must take into account the application of meta-heuristic and numerical optimisation algorithms to solve the same task and rewards. This will enable comparisons and conclusions to be drawn from the performance of DRL and non-DRL algorithms fairly.

Reinforcement learning environments require three main functions which are init, step and reset. The init space initialises the environment by defining all the necessary parameters and spaces. This must be followed by a step function which explains how the agent introduces its changes in the environment, defines the reward function and returns the new environment data (rewards, observations and more). Finally, a reset function is used to end the episode and initialise the next by resetting the parameters. In this section, we will cover all the methods used to create our custom environment. Additional functions are usually implemented for data manipulation and data visualisation purposes such as the render function. Reward abstraction

functions were used to allow the use of external optimisation algorithms in the training and testing stages. The full WDN-DRL environment code is listed in Appendix A.

4.2.1. Wrapping and Communicating with Epanet

Building the environment in a realistic manner requires an accurate representation of the hydraulic rules and properties of water distribution networks. To achieve this, we utilise the hydraulic prowess of EPANET. Whilst the environment should be written in python, EPANET does not have built-in API functionality. However, there are tools that allow users to interact with EPANET models through its toolkit. These tools are used for tasks such as model setup, simulation, and data retrieval. The EPANET Toolkit is a set of dynamic link libraries (DLLs) that allow the incorporation of EPANET's functionality into software applications. It allows programming languages such as C/C++ or python to create custom applications that interact with EPANET models. There are Python wrappers available for the EPANET Toolkit, making it easier to use EPANET through Python scripts. The most popular wrapper is called EPYNET which provides a Pythonic interface for creating, analysing, and simulating EPANET models (Vitens, 2017). This can be shown in **Figure 4-4**.

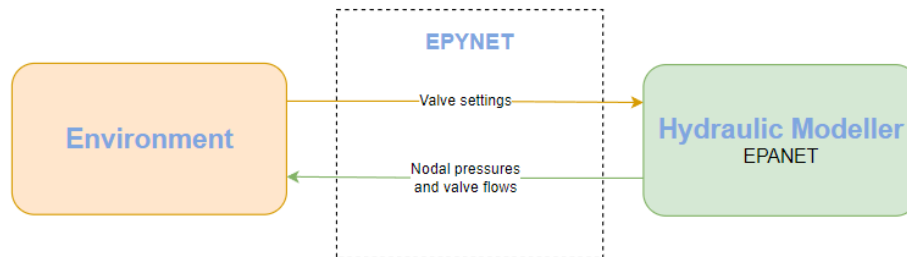


Figure 4-4 EPYNET wrapper communication schematic

The EPYNET wrapper allows the users to access all EPANET features through an object oriented pythonic interface, except chemical calculations which are irrelevant to the leakage problem.

Using this wrapper, it is possible to simulate steady state (single timestep) and extended period (multiple timestep) simulations. Since the leakage problem is highly dependent on varying customer demand patterns and temporal dependencies, it is necessary to evaluate extended period simulations (EPS). EPS runs a full 24-hour simulation using the provided hydraulic model except without an option to alter valve settings or pump speeds during the simulation. This is resolved by redefining the EPS simulation through many steady state simulations that read demand patterns and events iteratively. This method allows us to simulate the 24 timesteps for each hour of the day with an option of altering our actuators and

receiving new results. Having this agency can allow the user to also introduce leakage at different times and mitigate the leakage accordingly. Changing time parameters can allow the user to increase time steps to simulate up to each minute of the day, as long as the demand patterns used in the model can match these increments.

In **Figure 4-5**, flow rates for a node before introducing a leak (blue), after the leak (orange) and after decreasing the pressure in the node through valve action (green) are demonstrated. These events were introduced midday at the 24th timestep (each timestep being 30 minutes long). Experimenting with EPS through incremental steady state simulations proved successful as it permits action and evaluation throughout the simulation, which is necessary to create a real-time pressure optimisation strategy. **Figure 4-5** highlights how leakage influences the demand on nodes through an additional leakage flow rate. It also highlights how exploiting the pressure-leakage relationship (**Eq. 2-3**) can partially mitigate the effects of leakage through pressure management. In this case, we introduced a burst to a random node (J88) on the 12th hour of a 24hr extended period simulation.

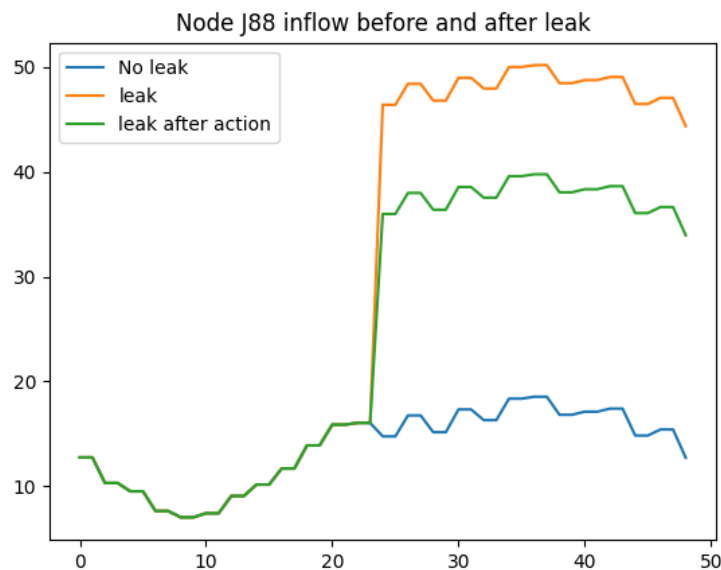


Figure 4-5 Introducing leakage and action using EPYNET.

It should be noted that the node’s inflow continues to follow the designated customer demand pattern with the minimum night flow present between the 8th and 10th timestep, signifying 4am to 5am. The EPYNET wrapper will be utilised in the environment building to harness the hydraulic functionalities of EPANET, harmonised with the availability of DRL benchmarking tool and algorithms in python programming.

4.2.2. Environment Spaces

Reinforcement Learning (RL) environments consist of various components and spaces that define the characteristics of the problem that an RL agent is trying to solve. For the purpose of availability and reproducibility, benchmarked spaces are used such as the OpenAI GYM. It is popular in the RL community for research, experimentation, and educational purposes. It has helped advance the field of reinforcement learning by providing a common platform for testing and comparing algorithms. GYM provides a framework for developing and testing deep reinforcement learning algorithms. It is designed with key test environments that are often used to compare and test DRL algorithms as a benchmark. It also allows users to create custom environments using its space to define the action space, and observation space. The power of writing environments using GYM's framework is its versatility. It can be integrated with wider RL libraries and other machine learning libraries such as TensorFlow and PyTorch.

Open AI GYM spaces consist of three main categories. The Box space (`gym.spaces.Box`) is a continuous space defined by a lower and upper bound. Each dimension can take on any value within the specified range. The bounds can be different for each dimension in the space or uniform throughout. Secondly, the discrete space (`gym.spaces.discrete`) represents a single dimension with a fixed number of possible discrete values. It is defined by a single parameter which denotes the number of possible discrete values. Finally, the Multi-Discrete (`gym.spaces.MultiDiscrete`) describes a multi-dimensional discrete space where each dimension can have a different number of possible discrete values. It is defined by an array of integers that specify the number of possible discrete values for each of these dimensions.

Other categories are available in GYM spaces including Multi-Binary (`gym.space.MultiBinary`) which represents a multi-dimensional space of binary arrays. Spaces such as Multi-Binary, Graph (`gym.spaces.Graph`) and Tuple (`gym.spaces.Tuple`) have limited use with external libraries. Consequently, they are less popular and incompatible with most available DRL agents. Understanding these different spaces is paramount for designing the environment and selecting DRL agents. The choice of the action and observation spaces must reflect the nature of the environment being designed and the specific problem being solved (pressure management for leakage minimisation). These spaces set the limits of what an RL agent can do, and they are crucial in shaping the agent's behaviour.

Observation Space

The observation state, also called state space, represents all the possible observations an agent can make in the environment. Accounting for the leakage problem, the observation

must describe the effect of the actions and leak on the consumption nodes. The leakage rates in each node were used to denote this. In addition, it is crucial to highlight the current settings implemented to serve as a starting point at each timestep. Hence, the observation space will describe the nodal leakage rates and actuator settings. Flow rates are outputted as floats by EPANET which means the leakage rates must be contained in a continuous space. The only available continuous space in GYM is the Box space which will be unbounded to allow the output of the leakage rate regardless of the value.

Action Space

The action space defines all the possible actions that the agents can perform within the environment. In the context of the leakage problem, it is important to allow the agent to send simultaneous orders to each actuator (PRVs or Pumps) to allow full control of the WDN during each timestep. Henceforth, the Discrete box cannot be used for this scenario. Another consideration is whether the new settings delivered to the PRVs, and pumps should be floats (continuous) or integers (discrete). Since this is treated as an optimisation strategy, the agent should be able to find the global maximum at the highest accuracy. This is representing using a Box space that allows the agent to select any value within the specified bounds. The action space will include as many dimensions as the valves and pumps being controlled with lower and upper bounds of 0 to 70 for valves and unbounded speeds for the pumps if applicable.

4.2.3. Step Function

The step function is arguably the most essential building block of any environment. It is a fundamental concept that defines how the agent interacts with the environment at each timestep. This step is enacted once every timestep iteratively until the conditions to terminate the episode is met. During each episode, the environment is designed to simulate 24h steps to model the 24-hour timesteps in the demand pattern used. This allows the agent to change each PRV simultaneously to match the change in demand for each hour. Repeatedly acting and observing the consequences through the step function formulates the RL learning process allowing the agent to develop a desirable policy that prioritises better decisions to maximise long-term rewards. Therefore, several tasks must be performed within the step function which are displayed in a flowchart in **figure 4-6**.

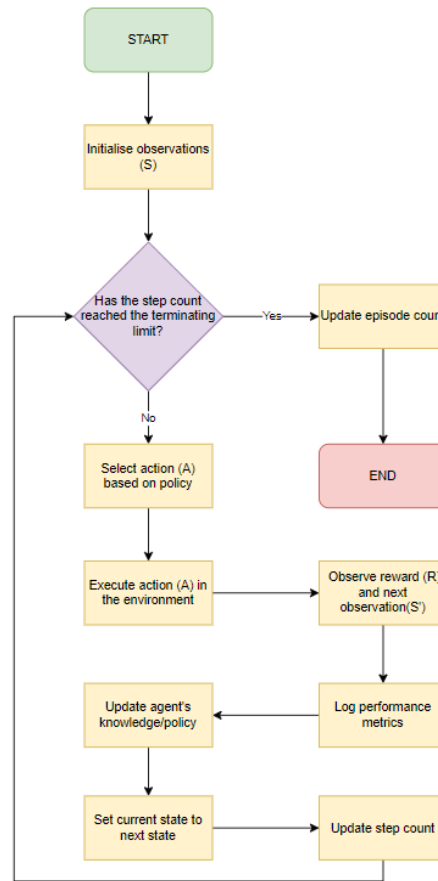


Figure 4-6 Step function flowchart

The step function must take the agent’s decision as an input. In DRL this decision stems from an input of the observation where the trained agent can choose the appropriate action from the predefined action space through its weighted neural network. This action must be executed on the environment through interactions that accurately reflect the task. These interactions must take the agent’s action as input and deploy it in order to update the environment’s state. In terms of the leakage problem, this action is an array of PRV settings or an array of PRV settings and pump speeds. These selected actions are assigned to their respective actuators through communication with EPANET and retrieve the resulting hydraulic properties of the water distribution system. In addition, the action must be assigned a numerical reward based on its perceived effects on the environment’s state and observations. Before the end of each iteration, important hydraulic results are logged for data visualisation and reporting purposes. Finally, the reward and observations must be returned as feedback to the agent, accompanied by a signal that indicates whether the episode has terminated.

Developing the step function for the leakage problem will require the hydraulic solver to evaluate the water network under three conditions. Namely, the network before the leakage is introduced (Perfect network), the network after the leakage is introduced (Leaking Network),

the leaking network after it is mitigated with the action (Solved Network). During each time step, the leaking and perfect networks are compared to demonstrate the initial effect of the leakage event, followed by contrasting the solved and leaking network to highlight the effect of pressure management on the two networks. The difference in performance between the solved and leaking networks will prove as a measure of the effect of the action on mitigating the leak. Further details on how this effect is calculated can be found in section 4.2.4 that defines the reward function.

4.2.4. Reward Control

In DRL, the reward is a term given to the numerical value that grades the agent's behaviour in its environment after taking an action. It is the primary feedback signal that represents how good or bad the agent was in fulfilling its objectives. The aim of this reward is to guide the DRL algorithm to develop optimal or near-optimal policies through maximising the cumulative reward. In other optimisation algorithms, the reward is replaced with a cost function, fitness value or a penalty which all denote the same concept. Understandably, the reward function has considerable influence on the training and development of DRL models. Therefore, designing an effective reward can be a significant task, and if poorly designed it can lead to suboptimal or unintended behaviour in the learned policies.

Reward formulation is an intricate task, and it should reflect the desired objectives. Rewards can be sparse or dense (assigned infrequently or more frequently). Despite sparse reward being easier to design, they can make learning challenging due to limited feedback. By contrast, dense rewards often accelerate training. A favourable reward can help agents navigate the exploration-exploitation dilemma by encouraging a good balance between learning new actions and choosing actions that maximise rewards. Another way to address this trade-off is through the discount factor (γ) which helps the agent prioritise short-term gains and long-term consequences. The reward signal is often processed by the DRL learning algorithm to develop the value function (V) or policy objective function (J), enabling it to make better decisions over time. Spurious rewards could promote unintended behaviour and force agents to converge at local minima or undesired policies. Therefore, rewards must follow reason and be subject to human monitoring. It's common to improve iteratively on the reward function as insights are obtained from the agent's training and performance.

After several iterations, the general formula for the reward function consists of the evaluation of penalties incurred by the burst events (P_{Leaking}) and the action provided by the optimisation algorithm (P_{Solved}). The reward is the difference between the two penalties to highlight how the

action decreases the negative effect of the burst. These penalties are denoted by **equation 4-3** which come together to form the reward (R) in **equation 4-4**.

$$R = \sum_{j=1}^M [(v_{after} - v_{before}) \cdot scale\ 1 + \left(\frac{Q_{lb} - Q_{la}}{Q_{lb}}\right) \cdot scale\ 2] \quad (4-3)$$

Where R is the reward for the current step, j is the current node, M is the total number of nodes, v_{after} is whether the nodal pressure at j is below 10m or above 70m after the mitigating action and the v_{before} is whether the nodal pressure at j below 10m or above 70m before the action. The upper and lower limits of 70m and 10m represent the limits put in place by the UK regulatory body OFWAT. The second half of the equation represents the effects of leakage rate on the reward function. Q_{lb} is the leakage rate at node j before the agent's action and Q_{la} is the leakage rate at node j after the action. This creates a fair assessment of each leak change as a proportion of its original leakage rate. The scales 1 and 2 are customer-defined integers defined through iterative training and testing of DRL algorithms. The scales must be tuned relative to each other to strike the best trade-off between the two objectives. P_{Leak} is the penalty calculated using **equation 4-3** by contrasting the leaking network (after) and the perfect network (before) whilst P_{Solved} evaluates the solved network (after) with respect to the perfect network (before).

The correct scales for the reward function are found iteratively for each case study by training the same DRL algorithm using different scales, testing all the algorithms under the same conditions, and plotting their performance in a three-dimensional plot. This plot highlights the trade-off explored by each set of scales with respect to water saved and violations minimised. By evaluating the plots with consultation from Northumbrian Water and Designed Network Solutions, the best set is selected. Below the performance of 13 identical A2C algorithms

trained using different reward scales. Investigating the ratios between the reward scales helps explore the trade-off between the two main objectives – leakage and pressure violations.

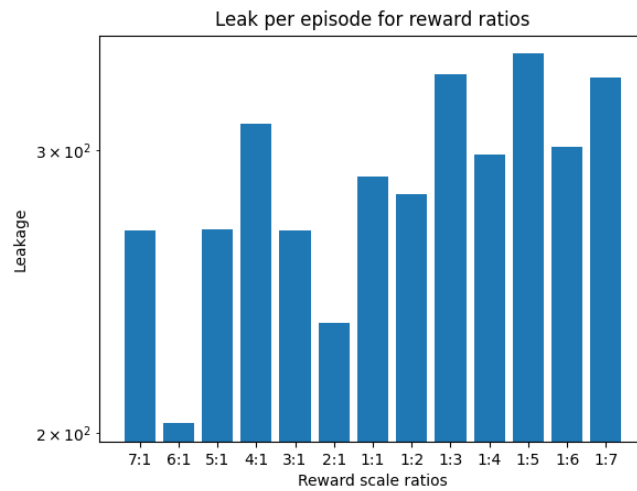


Figure 4-7 Episodic leakage rate vs reward scales

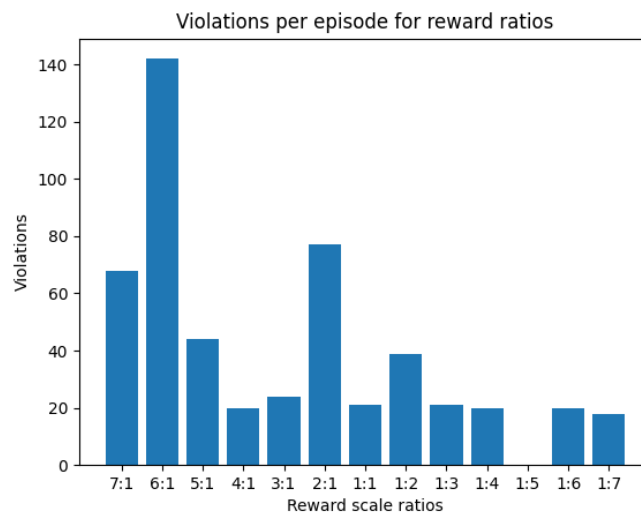


Figure 4-8 Episodic violations vs reward ratios

Figures 4-7 and 4-8 display how emphasising different objectives affect the agents' performance where a ratio of 6:1 (scale 2:scale 1) produced the best result in minimising leakage rate and a ratio of 1:5 produces the best result in minimising pressure violations.

Combining the two graphs help with the selection of the best overall reward scales ratio. This is displayed in **figure 4-9** below.

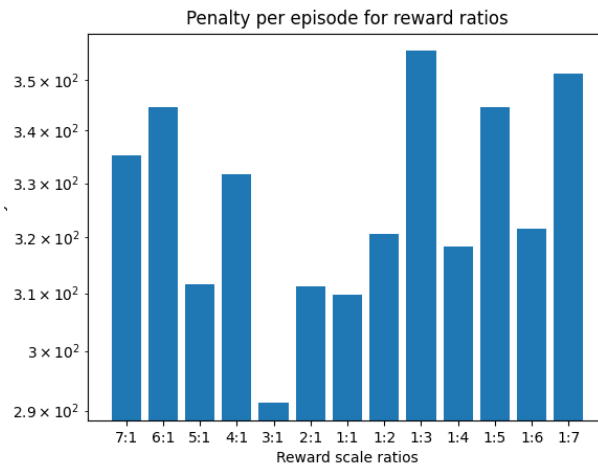


Figure 4-9 Episodic penalty vs reward scales

Clearly, the combination of the two objectives is best managed by placing a 3:1 emphasis on leakage reduction with respect to pressure violation. Furthermore, a 3D contour plot helps reveal an extra layer of time. **Figures 4-10 to 4-12**, show how the different agents interacted during each step of the episode. The contour plot is used to notice trends between reward formulation and agent behaviour.

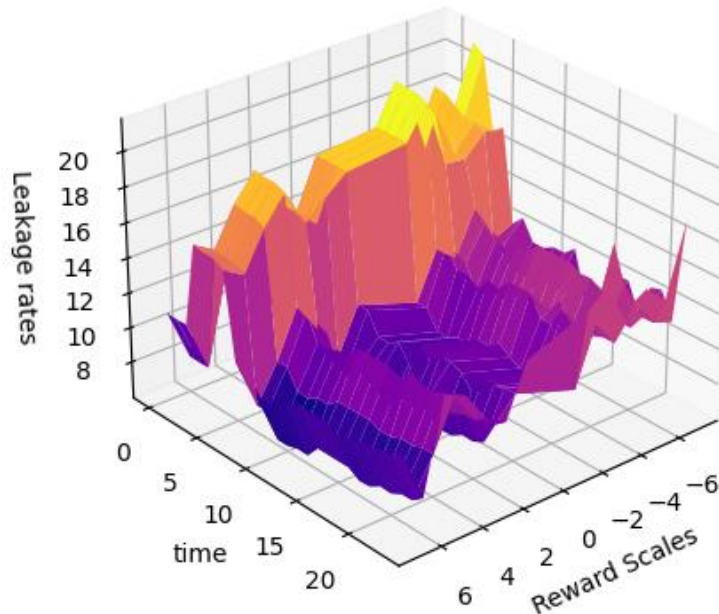


Figure 4-10 Leakage rate - reward scales 3D plot

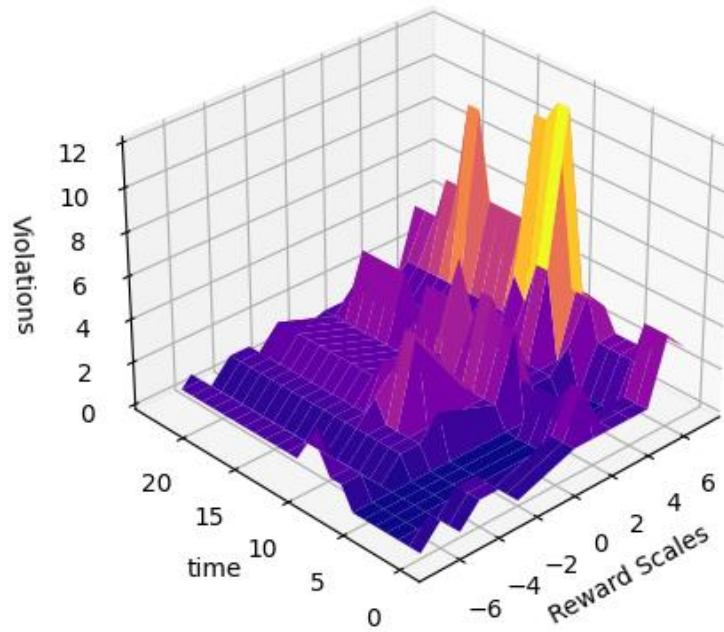


Figure 4-11 Violations - reward scales 3D plot

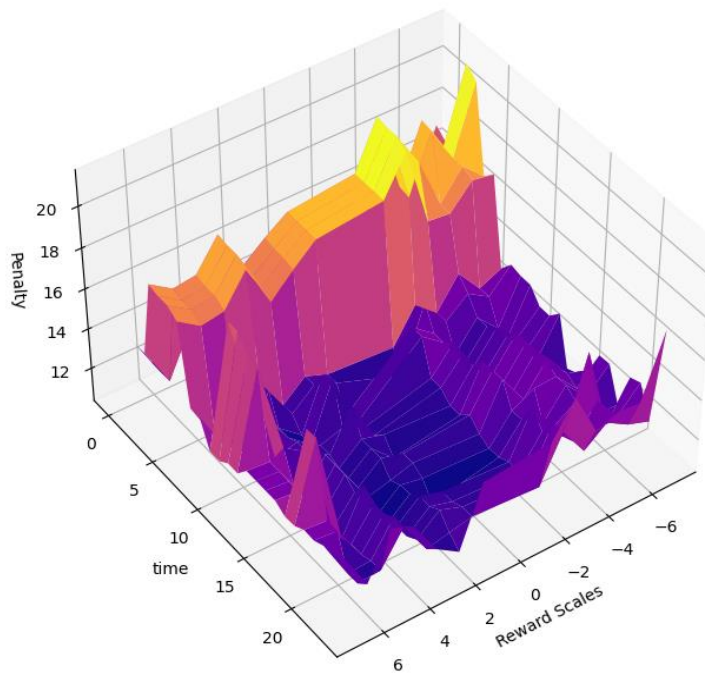


Figure 4-12 Penalty - reward scales 3D plot

The 3D plots identify a major vulnerable area between 3am and 6am due to minimum night flow and higher pressures. The higher pressures increase leakage during the low demand hours. It is also evident that extreme sides of the ratios (7 indicating 7:1; -7 indicating 1:7) don't necessarily produce the best results for the corresponding objective. Nevertheless, emphasising the leakage objective has clearly produced the best overall results.

4.2.5. Render Function

Understanding the performance of the DRL agents provides more than a testing platform, it can inspire improvements in the environment design and contextualise the results within the leakage problem. The render function is not a critical function in reinforcement learning environments. However, it is commonly used to visualise the interactions between the agent and the environment. Displaying this data in a digestible manner can aid in diagnosing any issues in the agent's behaviour. It can also help with fine-tuning the reward function and the agent's hyperparameter as it provides more feedback on the agent-environment interactions. Further analysis can be conducted between learnt policies by drawing comparisons from their rendered visualisations. Rendering can incur additional computational costs, so the render function is best reserved in the testing loop of DRL algorithms. In summary, rendering is a useful tool for understanding, debugging, analysing, and tuning DRL algorithms.

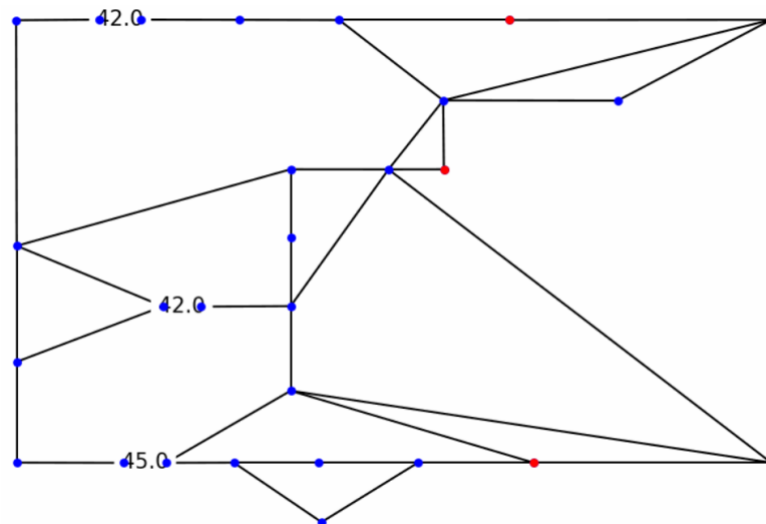


Figure 4-13 Example of the interactive map render.

The environment was equipped with five unique rendering functions that highlight the main aspects. The first rendering function is called the 'interactive map' and draws the WDN architecture in full, as shown in **Figure 4-13**. The leaking nodes are coloured red, normal nodes are coloured blue, links are coloured green, and valves are replaced with their current settings. When the function is called, the environment records the map at each timestep with the settings displayed. It then collated them in a video to display how the settings change

throughout the day as well as the locations of the leak nodes.

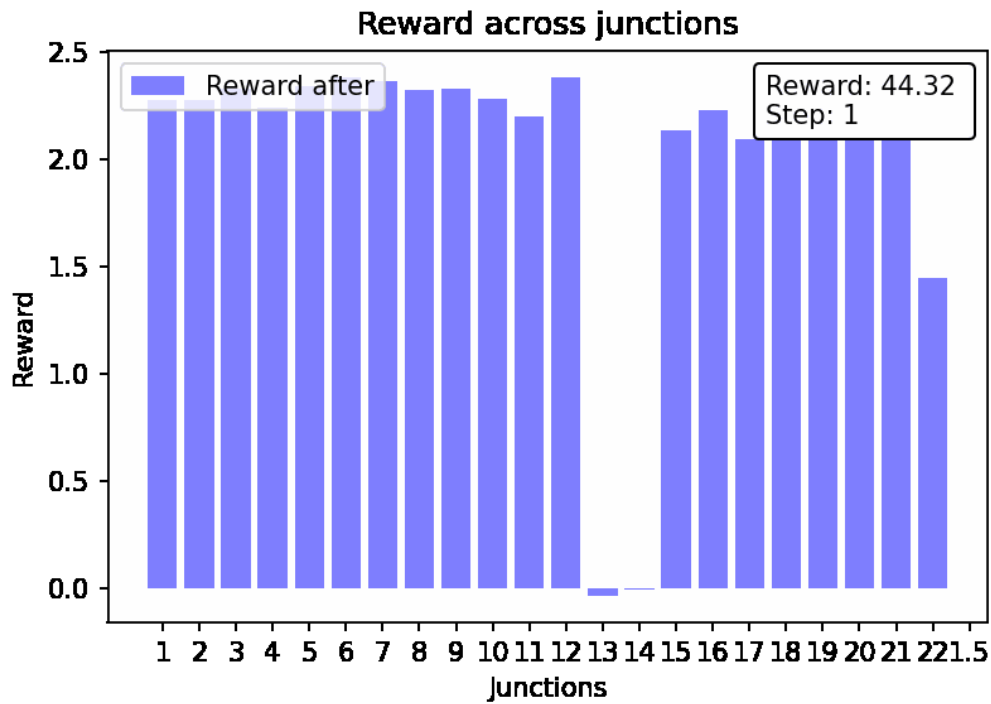


Figure 4-14 Example of reward spread across junctions render.

Another video rendering function is the 'penalty across junctions' option, shown in **Figure 4-14**, which displays the penalties before (P_{Leak} , red) and after (P_{Solved} , blue) the agent's action, spread across the network's junction for each step of the episode. This allows the user to visualise how the rewards are distributed across the network and how the agent's action affects the network. Rewards with a value of 1 are clear indications of resolved pressure violations as shown in **Figure 4-14**. Whilst larger rewards can be seen where leakage has been reduced due to valve action. These figures highlight the learnt policy and where the agent has placed emphasis to maximise its reward.

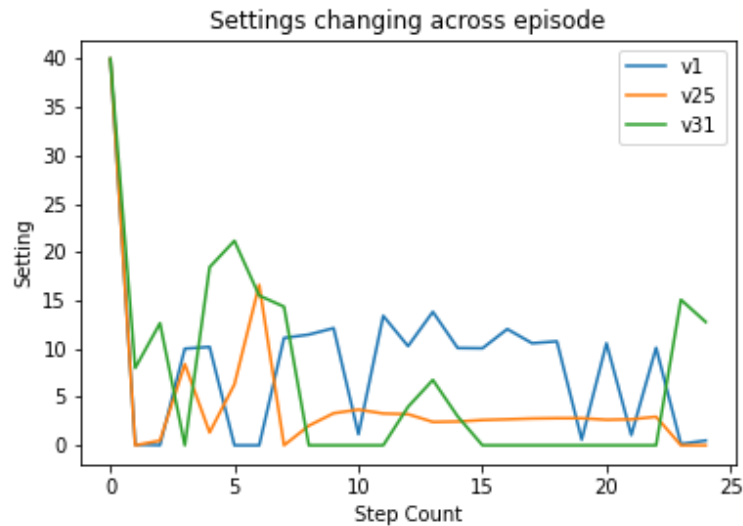


Figure 4-15 Example of settings render.

The 'settings' render function, shown in **figure 4-15**, records the agent's selected actions throughout the episode to provide an agent's point of view. This display can highlight whether the agent has overcome the exploration-exploitation dilemma. If the agent shows no changes in the PRV settings, then it is possibly exploiting a local minimum and has not learnt an optimal behavioural policy. Whereas overly stochastic behaviour coupled with low rewards might indicate an overly explorative policy. If the agent is able to produce similar 'settings' renders under low disturbance and different renders under high disturbances, it is clear that it has learnt a complete policy.

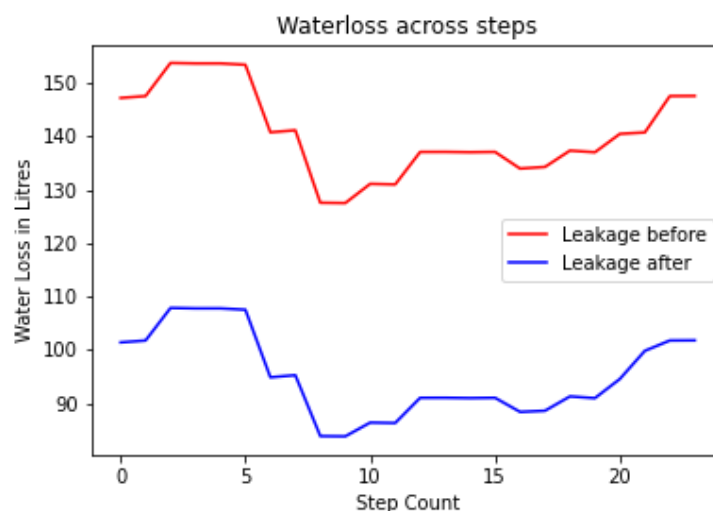


Figure 4-16 Example of Water Loss render.

The ‘Water Loss’ render, shown in **Figure 4-16**, is another plot that displays leakage rates from the leaking network in red (leakage before action) and the solved network in blue (leakage after action) during the episode. This highlights the hydraulic effects of the pressure management algorithms on leakage in the network. It is important to note that both plots must show similar patterns that are indicative of the customer’s demand patterns. The variance in magnitude highlights the effect of the agent’s action and whether it has minimised water loss.

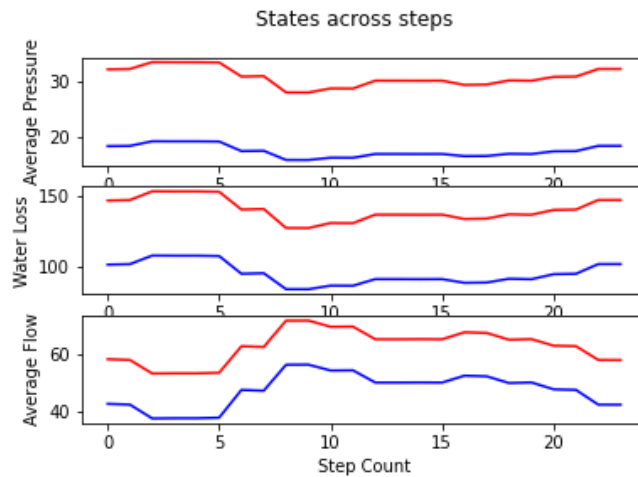


Figure 4-17 Example of the states render.

Finally, the ‘states’ render displays more hydraulic data logged during the episode. As shown in **figure 4-17**, the state’s visualisation plots three main graphs that show the average pressure, the total water loss, and the average flow of the WDN in three subplots respectively. In each subplot, the user can visualise how the actions have affected these properties to draw more conclusions. It is beneficial to see the agent minimise fluctuations in the average pressure as it strives to minimise nodal pressures at leakage nodes.

Further visualisation tasks may be necessary such as plotting the rewards gathered during the episode hence why a report function was also designed. The report function exports all the logged metrics to an external excel file. This includes the observations, water loss, flows, pressures, violations, and penalties of each junction as well as the current observation, and list of valve settings for each step of the episode. Furthermore, a summarised spreadsheet of states and rewards of both the leaking and solved networks is coupled with a full history of settings for each episode. These reports are used for detailed analysis and conclusion of agents’ performances.

4.3. The Agents

Deploying agents is simple using a well-written environment that follows OpenAI GYM's formalism. Several libraries have customisable pre-built DRL agents that are deployable in GYM-based environments such as Stable Baselines 3 (SB3), and Keras RL. Otherwise, agents must be built from scratch using TensorFlow and PyTorch. These libraries include state-of-the-art algorithms such as Advantage Actor Critic (A2C), Proximal Policy Optimisation (PPO), Deep Deterministic Policy Gradient (DDPG) and more. The most extensive library is the Stable Baselines 3 which is a set of improved implementations of DRL agents based on OpenAI Baselines. It features a unified structure for many algorithms boasting visualisation tools, Tensorboard logging and thorough documentation. The only limitation in agent selection is compatibility with the action space. **Table 4-2** outlines the different algorithms available and their compatibilities. The action space used in the WDN-DRL Ecosystem is a continuous (Box) space meaning that the available agents are Augmented Random Search (ARS), A2C, Hindsight Experience Replay (HER), PPO, Recurrent PPO, Soft Actor Critic (SAC), Twin Delayed DDPG (TD3), Truncated Quantile Critics (TQC), Trust Region Policy Optimisation (TRPO), DDPG, and Normalized Advantage Function (NAF) as shown in **table 4-2** below.

Table 4-2 Available DRL agents

Abbreviation	Name	Box	Discrete	MultiDiscrete	Library
ARS	Augmented Random Search	✓	✓	✗	SB3
A2C	Advantage Actor Critic	✓	✓	✓	SB3
DDPG	Deep Deterministic Policy Gradient	✓	✗	✗	SB3
DQN	Deep Q Network	✗	✓	✗	SB3
HER	Hindsight Experience Replay	✓	✓	✗	SB3
PPO	Proximal Policy Optimisation	✓	✓	✓	SB3
QR-DQN	Quantile Regression DQN	✗	✓	✗	SB3
Recurrent PPO	Recurrent Proximal Policy Optimisation	✓	✓	✓	SB3
SAC	Soft Actor Critic	✓	✗	✗	SB3
TD3	Twin Delayed DDPG	✓	✗	✗	SB3
TQC	Truncated Quantile Critics	✓	✗	✗	SB3
TRPO	Trust Region Policy Optimisation	✓	✓	✓	SB3

Maskable PPO	Maskable Proximal Policy Optimisation	✗	✓	✓	SB3
DQN	Deep Q Network	✗	✓	✓	Keras RL
DDPG	Deep Deterministic Policy Gradient	✓	✗	✗	Keras RL
NAF	Normalized Advantage Function	✓	✗	✗	Keras RL
CEM	Cross-Entropy Method	✗	✓	✓	Keras RL
SARSA	State-Action-Reward-State-Action	✗	✓	✓	Keras RL

Importantly, through testing and comparisons, the best performing algorithms are selected for the case studies. The agents are trained for 20,000 timesteps on the environment created. Since we create our scenarios from the hydraulic files on epanet, our minimum data requirements were the demand patterns, node lengths and GIS data needed to create the input file.

Furthermore, various neural network architectures will be tested and hyperparameter tuning unveils the best possible version of the selected agents. In this section we explain the agents used in the case studies which produced the best results. The agents used fall into three main categories of hybrid, policy-driven, and distributional. The full code used to train the DRL algorithms, save the models and deploy them in the test scenario is shown in Appendix C.

4.3.1. Hybrid DRL Agents

Advantage Actor Critic (A2C)

A2C is an on-policy model-free DRL algorithm where two neural networks communicate to improve the algorithm's performance. The actor's neural network follows a policy-driven method to produce an action which will be evaluated using the value-driven critic neural network. Whilst the actor dictates how to act in a method similar to the PPO algorithm, the critic evaluates how well that action was in a method similar to the Deep Q-Network (DQN). The actor is influenced by the critic through a temporal difference (TD) error loop. This is shown in the schematic in **figure 4-18**.

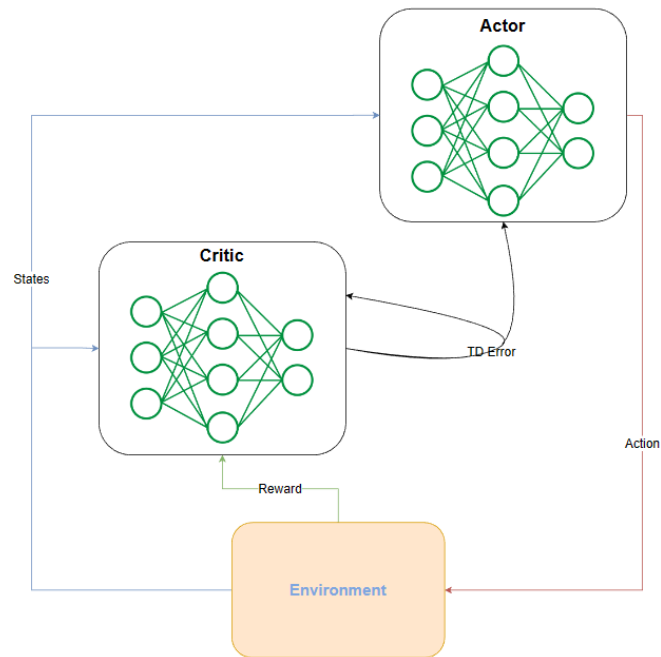


Figure 4-18 Advantage Actor Critic model schematic

This helps in creating two diverse experiences of acting and critiquing to enhance the overall learning process. In this algorithm, we use identical setups to create the agents. A key concept in A2C is the use of the advantage function (Eq. 3-9) in the critic network rather than the original value function. The advantage function evaluates the action's benefits relative to the average action. Doing so, A2C reduces variance in the learning process leading to more stable and efficient learning.

The agent's neural network architecture consists of two main sections. The function approximator is followed by the network architecture as displayed in **figure 4-19** below. The observation states (nodal pressures and valve settings) pass through a feature extraction class to abstract the input data. This is then fed into a fully connected deep neural network with an architecture of two hidden layers of 64 neurons each. Each layer is equipped with a tanh activation function and an Adam optimiser class that lead to the output layer with the size of the action space. The feature extractor is named the 'Nature CNN' which comprises of a convolutional neural network first deployed in (Mnih *et al.*, 2015). This method consists of three hidden convolutional neural network layers. The first layer has the size of the observation space x 32 with a stride of 4 and kernel size 8 followed by the second layer of size 32x64 with a stride of 2 and kernel size of 4 and finally the third layer takes the shape of 64x64 with a stride of 1 and kernel size 3. All three layers use a Rectified Linear Unit (ReLU) activation function, and the final output is flattened to be processed by the network architecture.

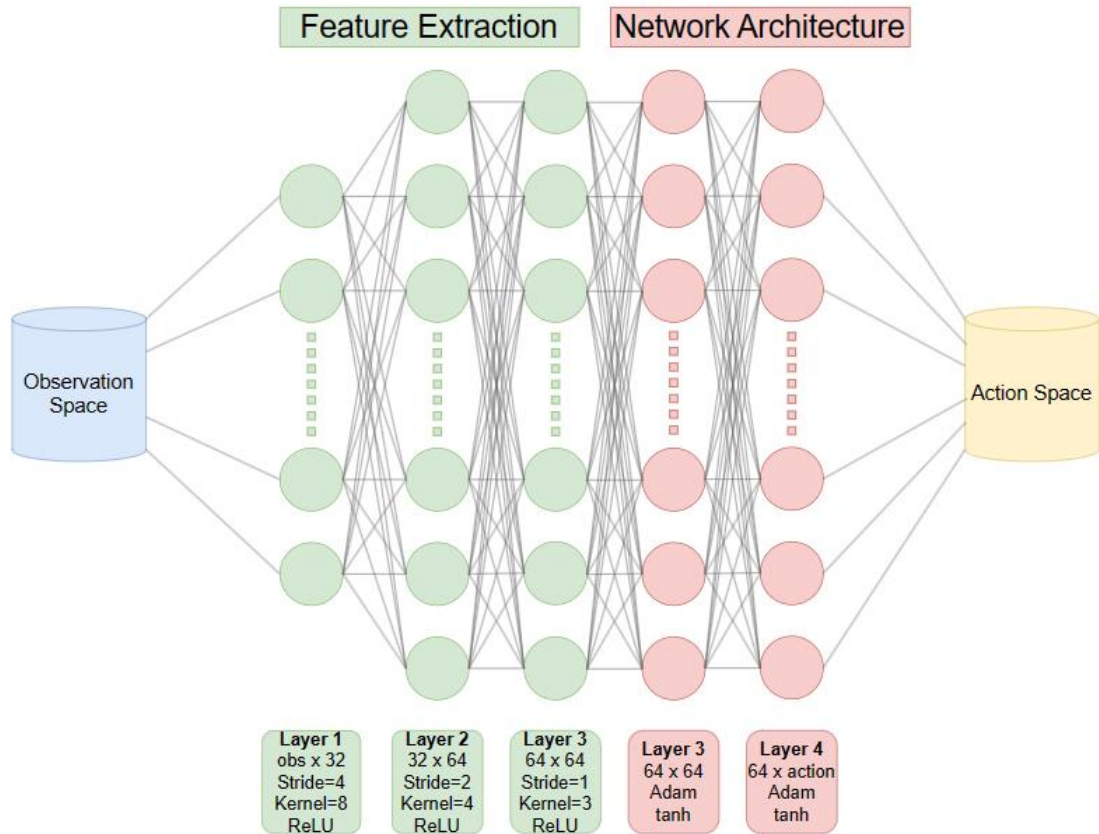


Figure 4-19 A2C Policy Network Diagram

Deep Deterministic Policy Gradient (DDPG)

DDPG is an off-policy hybrid DRL agent that serves as the alternative to deep Q-learning designed for continuous action spaces. On the critic's side, exhaustively computing Q values for each action in a continuous space is too computationally expensive. By assuming that the state action value (Q-value) is differentiable with respect to the action, DDPG sets deterministic policy that exploits this fact and reduces the computational load significantly. This is done using the mean squared bellman error (MSBE) shown below (eq. 4-5) Lillicrap *et al.*, 2016).

$$L(\theta, D) = \mathbb{E}_{(s,a,r,s',d) \sim D} [(Q_{\theta}(s, a) - (r + \gamma \cdot \max_{a'} Q_{\theta}(s', a')))]^2 \quad (4-4)$$

Where $Q_{\theta}(s, a)$ is the predicted state-action value, D is a collected set of transitions, d is the terminal state. The immediate reward is denoted as r, γ marks the discount factor, and the expression $\max_{a'} Q_{\theta}(s', a')$ is the expected value of the maximum next state-action's value.

L is the loss function which comprises of the square of the expected state-action value ($Q(s,a)$) minus the sum of the reward and the discounted expected next state value $E_{s'}[V(s')]$. Q learning algorithms for function approximators often rely on MSBE minimisation. In this case

we use a replay buffer to build the transition (D) set using memories of previous experiences. The size of the replay buffer effects the learning process where having a small buffer causes overfitting and large buffers could include outdated experiences.

On the actor's side, DDPG builds a deterministic policy (π_{θ}) using the assumption that the Q value is differentiable with respect to the action. This is achieved by performing gradient ascent to solve the policy parameters to maximise the expected Q value. DDPG is known for its high convergence properties yet building deterministic values can be sub-optimal for real life engineering applications. DDPG is implemented using the SB3 library which uses the following deep neural network formed of three layers of convolutional neural networks (CNN) for feature extraction and two fully connected layers for policy development as shown in **figure 4-20**. The hidden layers of the feature extractor contain three CNN layers with 32 nodes, 64 nodes and 64 nodes. The kernel size for the first layer is 8 with a stride of 4, the second layer has a kernel size of 4 and a stride of 2, and the third layer with a kernel size of 3 and a stride of 1. The fully connected hidden layers have 400 nodes and 300 nodes respectively using Adam optimisers and tanh activation functions.

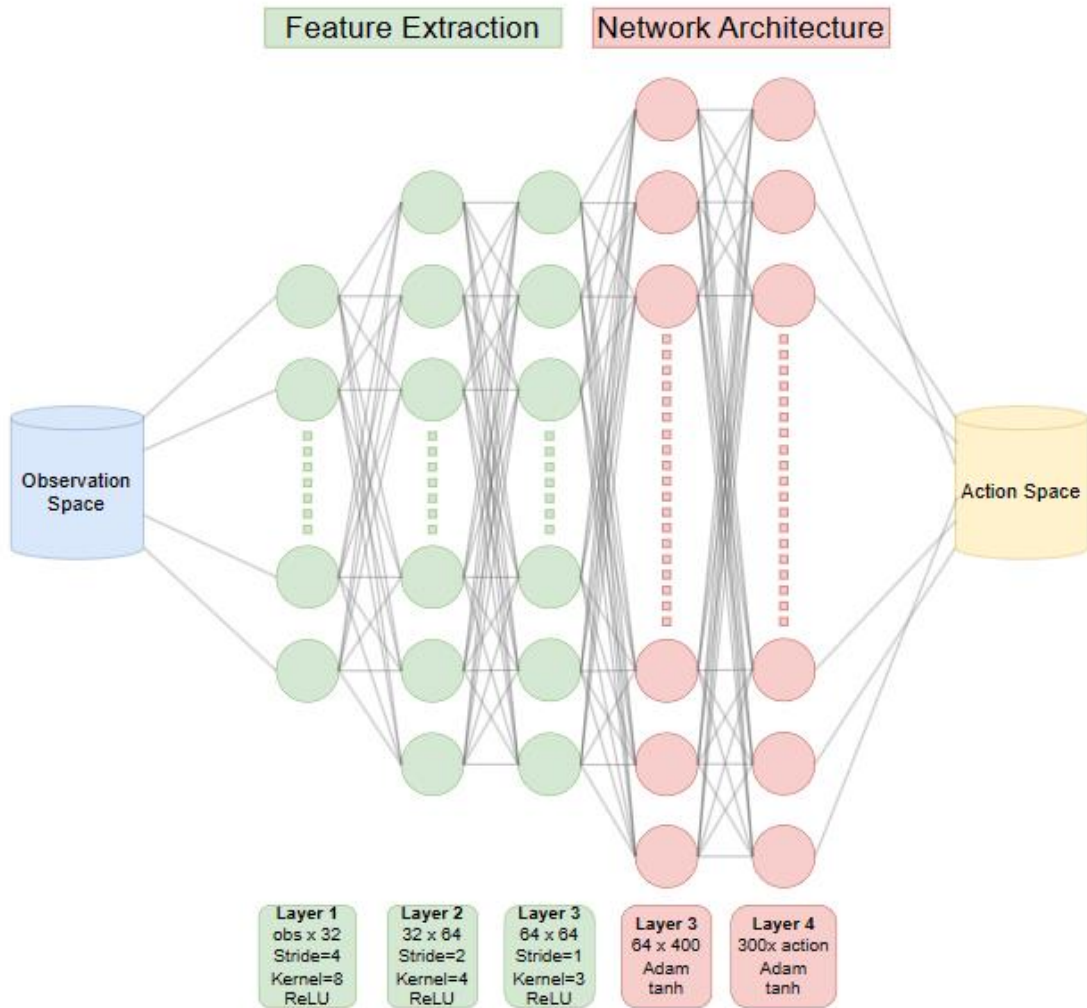


Figure 4-20 DDPG policy network architecture

Soft Actor Critic (SAC)

SAC is an off-policy model-free DRL algorithm that improves on DDPG's format by introducing stochasticity and entropy regularisation. Including the entropy in the objective function encourages stochasticity in the policy optimisation. This is achieved by altering the objective function in the critic to include entropy (H) as follows:

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P} [R(s, a, s') + \gamma(Q^\pi(s', a') + \alpha H(\cdot | s'))] \quad (4-5)$$

Where $Q^\pi(s, a)$ is the current quality function using the current state-action pair (s,a), $R(s, a, s')$ is the expected return based on the state (s), action (a), and the next state (s'). The next state and action are derived from the transition probability ($s' \sim P$) and the current policy ($a' \sim \pi$). The entropy is introduced as a bonus to the value function ($H(\cdot | s')$) and regularised using the discount factors (γ, α) (Soft Actor-Critic — Spinning Up documentation, no date).

The stochastic improvement helps the hybrid model navigate the exploration-exploitation trade off by rewarding entropy and penalising overly deterministic policies. Therefore, SAC is better fit in optimising tasks that require effective exploration. Like A2C, the soft actor critic algorithm deploys an actor (policy) and critic (value function) networks qualifying it as a hybrid method. The neural networks connect three layers of a function approximating CNN network to the observation space. This feature extractor class consists of a 32-node layer with a stride of 4 and a kernel size 8 followed by a second layer with 64 nodes, a stride of 2 and a kernel size of 4. The final layer also consists of 64 nodes, a stride of 1 and kernel size of 3. The feature extractor then feeds to the main network architecture that consists of two fully connected layers consisting of 256 nodes each. The feature extractor utilises a rectified linear unit activation function while the network architecture uses a tanh activation function and an 'Adam' optimiser class. The final layer is connected to the action space to produce a relevant output. This deep neural network schematic can be seen in **figure 4-21** below. The larger neural network is likely to increase computational demand in comparison to A2C yet is still an improvement to the DDPG neural network computational load.

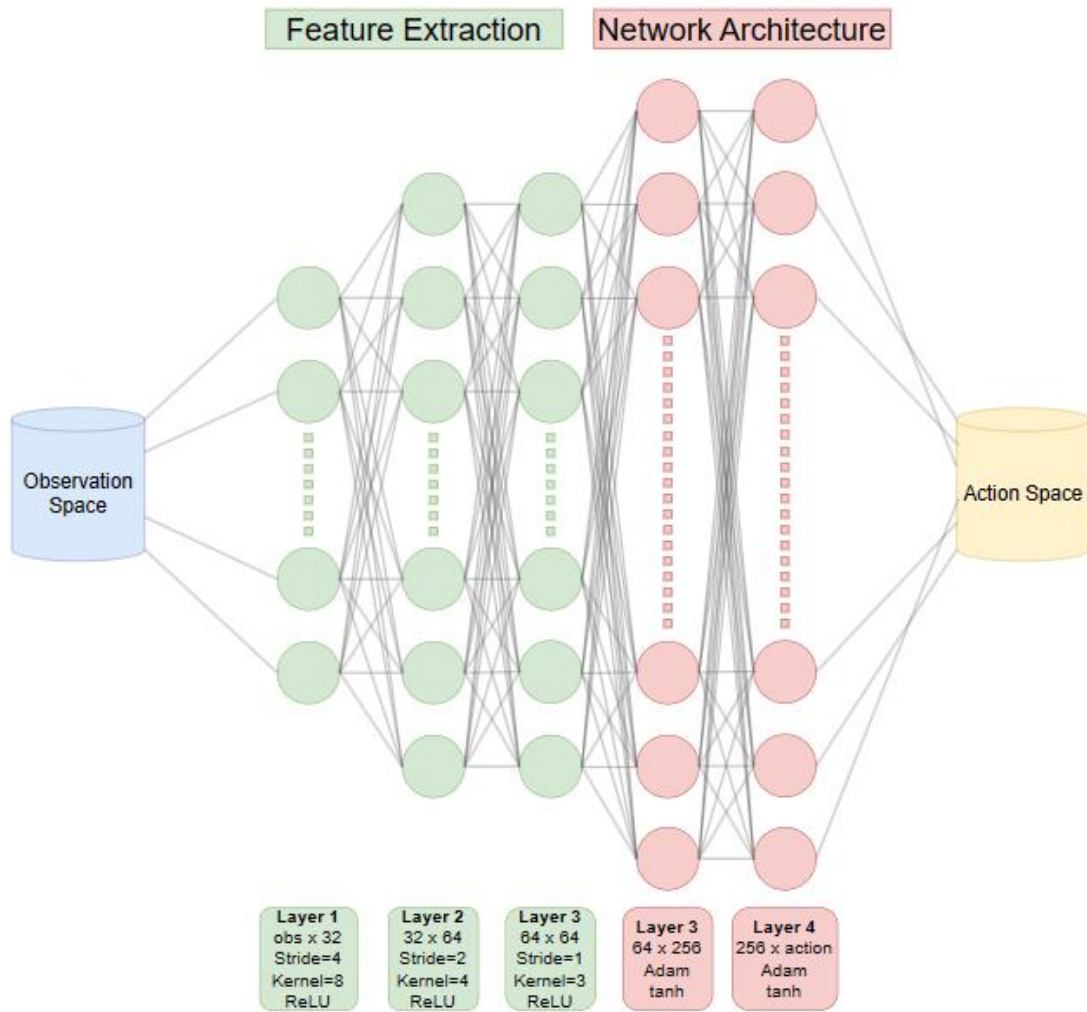


Figure 4-21 SAC policy network diagram

4.3.2. Policy Driven DRL Agents

Trust Region Policy Optimisation (TRPO)

In the search for robust performance and monotonic improvements, researchers have developed a policy gradient method through theoretically justified approximations – Trust Region Policy Optimisation. TRPO is an on-policy gradient method suitable for both continuous and discrete action spaces. Unlike normal policy gradient methods that keep policy updates close in the parameter space, TRPO takes the largest step possible to improve performance while satisfying a KL-divergence constraint that limits how close policy updates can be. In vanilla policy gradients, small changes in the policy updates can lead to very different performances meaning that bad steps can collapse policy performance completely. However, TRPO’s monotonic improvements improves sample efficiency and overall reliability by tackling the sensitivity in performance appearing from small updates in the policies parameters (*Trust*

Region Policy Optimization — *Spinning Up documentation*, no date; Schulman *et al.*, 2014).

This is achieved using the equations below that dictate policy updates.

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} L(\theta_k, \theta) \text{ such that } D_{KL}(\theta || \theta_k) < \delta \quad (4-6)$$

Where the next policy parameters θ_{k+1} , is defined using the best policy parameters of the surrogate advantage $L(\theta_k, \theta)$. The surrogate advantage is a measure of how the policy π_θ performs with respect to the old policy π_{θ_k} . The surrogate policy is dictated by **equation 4-8** below. In addition, this parameter update must satisfy the condition where the average KL divergence between policies across visited states $D_{KL}(\theta || \theta_k)$ remains below the pre-defined KL-divergence limit δ .

$$L(\theta_k, \theta) = E_{s, a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a) \right] \quad (4-7)$$

Where the surrogate advantage is derived from the expected advantage values of the old policy $A^{\pi_{\theta_k}}(s, a)$ multiplied by the ratio of the current policy divided by the old policy $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$.

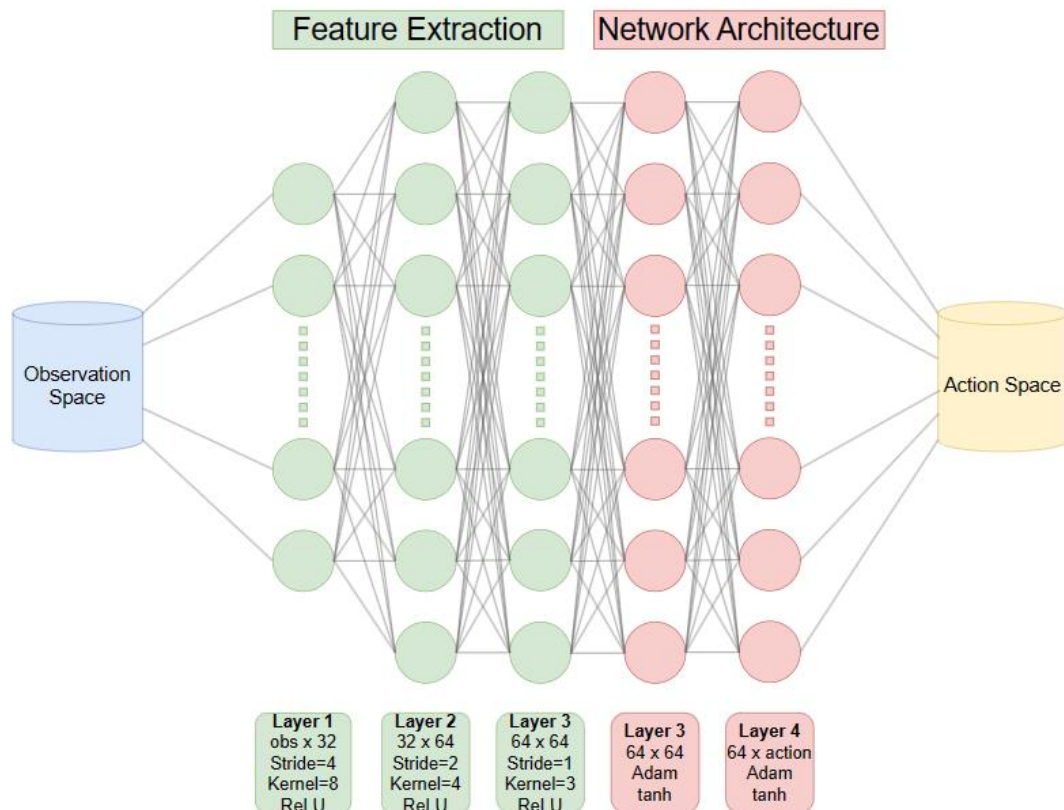


Figure 4-22 TRPO policy network architecture

The neural network architecture used to develop this algorithm is the multi-layer perceptron neural network shown in **figure 4-22** and is described as follows. The network architecture

consists of an input layer (the observation space), five hidden layers and the output layer (the action space). The hidden layers are trained to develop the algorithm’s policy; three of which are dedicated to feature extraction. The feature extraction neural network is the ‘Nature CNN’ consisting of three layers with 32 nodes, 64 nodes and 64 nodes respectively. The first CNN layer (32 nodes) has a kernel size of 8 and stride of 4 while the second layer has a kernel of 4 and stride 2 and the third has a kernel size 3 and stride 1. All the feature extraction CNN layers utilise a ReLU activation function and an Adam optimiser. The rest of the policy neural network architecture is marked in red and involves two fully connected layers with 64 nodes each, an Adam optimiser, and a tanh activation function. This then feeds into the algorithm’s action space as the output layer.

Proximal Policy Optimisation (PPO)

First introduced by Schulman et al. (Schulman *et al.*, 2017), proximal policy optimisation describes algorithms that utilise policy gradient methods which alternate between optimising a surrogate objective function and sampling data through environment interaction. PPO has gained popularity due to its stability, ease of implementation and effectiveness in training DRL agents. It attains the data efficiency and reliability of trust region policy optimisation algorithm (TRPO) using first order optimisation hence improving sample efficiency and reliability. PPO uses surrogate objectives such as Kullback-Leibler (KL) clipping and penalty (adaptive or flexible) to improve performance. The PPO variant with KL penalty updates the policy parameters similarly to its predecessor TRPO by penalising the KL divergence in the objective function rather than having it as a hard constraint. On the other hand, KL clipping has no constraints, however it limits KL divergence through specialised clipping. This clipping surrogate objective improved performance greatly and resulted in higher gains than high performing algorithms such as Advantage Actor Critic (A2C) (Schulman *et al.*, 2017).

In the application of WDN pressure management, we utilise the PPO-clip algorithm using a Multi-Layer Perceptron (MLP) neural network to optimise the agent. The algorithms use a Multi-Layer Perceptron architecture to optimise the parameters of the policy through gradient ascent. The policy gradient can be defined as the gradient of the objective function J in **equation 4-9**.

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right] \quad (4-8)$$

Where the policy gradient is the gradient of the log of the parameterised policy ($\nabla_{\theta} \log(\pi_{\theta}(a_t | s_t))$) multiplied by the expected sum of returns ($R(\tau)$) for timesteps (t) in episode length (T). KL clipping will be optimised along with hyperparameter tuning to improve the agent’s training

and performance. The MLP neural network is identical to that of the TRPO policy network shown in **figure 4-22**. This consist of 3 CNN layers for feature extraction and 2 fully connected layer (FCN) for policy development.

Recurrent Proximal Policy Optimisation (Recurrent PPO)

Recurrent PPO is a model-free policy-driven deep reinforcement algorithm. It is an extension to the well-known PPO algorithm that enables it to use Long Short-Term Memory (LSTM) neural networks. Due to this simple change in neural network architecture, Recurrent PPO performance has been heightened in applications that require agents to recognise patterns in long-term dependencies. Hence, Recurrent PPO outperforms it predecessor in WDN applications that require agents to understand the delayed consequences of current actions (Pleines *et al.*, 2022).

The original MLP architecture used is identical to that of the PPO and A2C algorithms except the Recurrent PPO algorithm includes a LSTM module that consists of 1 layer of 256 nodes (**figure 4-23**). LSTM nodes add a recurrent feature to the neural network allowing it to recognise recurring patterns.

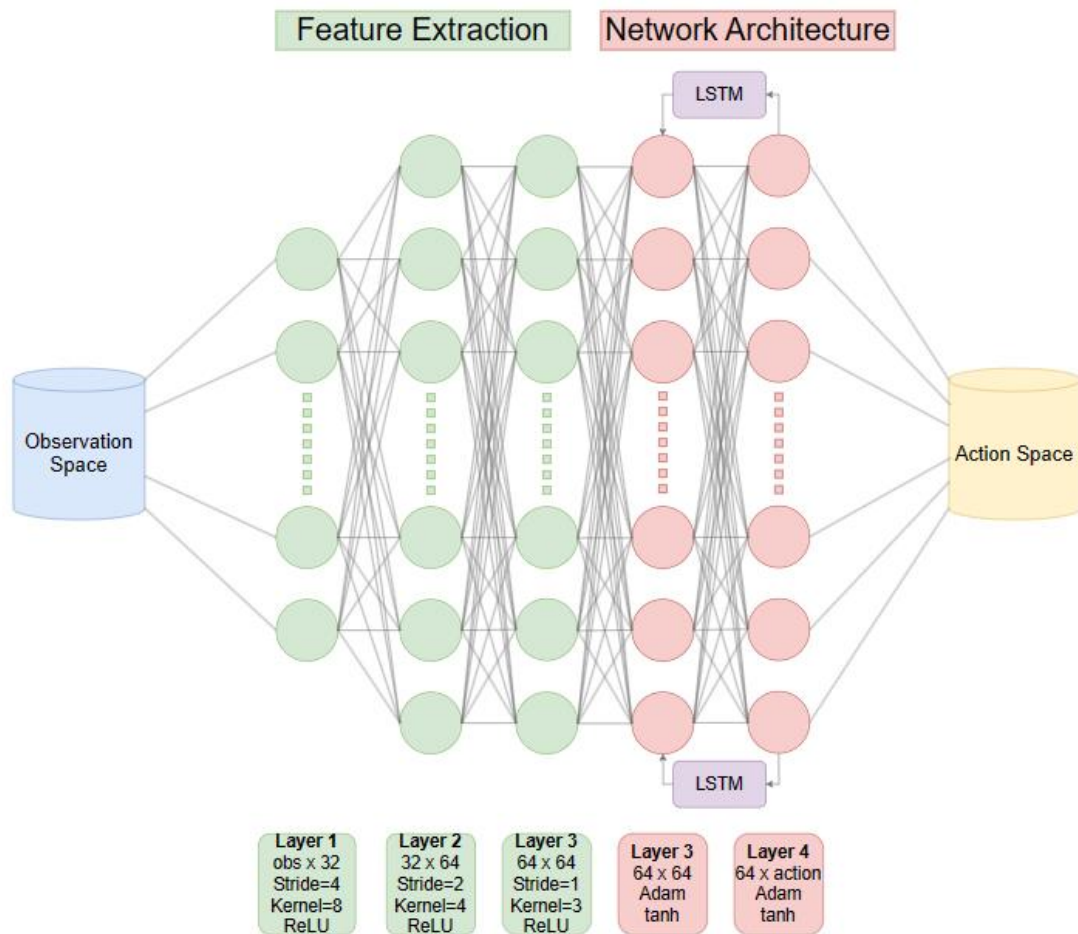


Figure 4-23 Recurrent PPO policy network architecture

Augmented Random Search (ARS)

ARS is a model-free DRL algorithm that utilises a simple random search algorithms with a few augmentations on the parameterised policy equation. This was first introduced in (Mania, Guy and Recht, 2018) where the authors attempted to improve sample efficiency in DRL models by developing a method that harnesses simple random search. ARS was aimed to present the simplest model-free method that can tackle current continuous control benchmarks in a more sample efficient manner than (Salimans *et al.*, 2017)'s evolution strategy (ES). Several policies are created through random perturbations that are ranked depending on their performance. The policy is updated based on the reward-weight sum of these perturbations. The adaptive step size controls the magnitude of these policy updates effectively tackling the exploration-exploitation dilemma. This process is repeated until the algorithm converges to the best achievable policy. Results from the initial experiment show an improvement in sample efficiency greater than 15 times than the best competing model-free methods and performed routinely higher than standard PPO, A2C, and TRPO algorithms (Mania, Guy and Recht, 2018). However, the ARS performance is plagued with high variance which suggests that the

estimations of sample efficiency don't represent the performance of the RL algorithm adequately (Mania, Guy and Recht, 2018). The ARS algorithm can deploy a linear policy (without the use of a deep neural network architecture) or a MLP policy that uses a neural network architecture identical to that of PPO and TRPO algorithms with three CNN layers and two FCN layers. This makes it possible to experiment with ARS as reinforcement learning algorithm rather than a deep reinforcement learning algorithm. ARS has shown its ability to perform well with and without the use of deep learning as shown in (Mania, Guy and Recht, 2018).

4.3.3. Distributional DRL Agent

Truncated Quantile Critics (TQC)

TQC is a distributional model free off-policy DRL algorithm designed specifically to tackle the overestimation bias present in most off-policy algorithms. Overestimation bias is a phenomenon particularly present in algorithms that use function approximation such as neural networks. It denotes the constantly higher estimated value for state-action pairs than their true value which would lead to algorithms converging at suboptimal policies. Using function approximators introduces a degree of uncertainty in the learning process hence amplifying this overestimation bias. TQC innovates overestimation control by incorporating aleatoric uncertainty using truncated quantiles. The truncated quantiles improve on the value distribution by allowing the algorithm to prioritise the important region of the distribution and neglect the less relevant regions. On the other hand, traditional distributional DRL that represent the mean value distribution TQC also decouples the function approximators from the overestimation control allowing the use of multiple neural networks to be ensembled in a novel manner. TQC derives its stochastic policy updates from the SAC algorithms by incorporating entropy to the policy function. Furthermore, it draws from QR-DQN's distributional methods to form its quantile critics.

The TQC schematic consists of three function approximators (deep neural networks) dictating the actor, which determine the next action through entropy regularised policy updates; the critic network, which estimates the state-action pair quantile value distributions; and the critic target network, which is used to find discrepancies between the estimated value and true value of the state-action pair. The discrepancy between the critic and the target critic is represented by a Huber loss function that should be minimised by improving the critic's performance. TQC is ideal for environments with complicated and varied reward structures. In this research, we utilise deep neural networks as function approximators for the actor, critic, and target critic. These neural networks involve three layers of feature extraction using CNNs

and two FCN layers as shown in **figure 4-24** below. All the layers use an Adam optimiser. The activation functions used for the CNN layers is the ReLU function while the FCN layers use a tanh activation function.

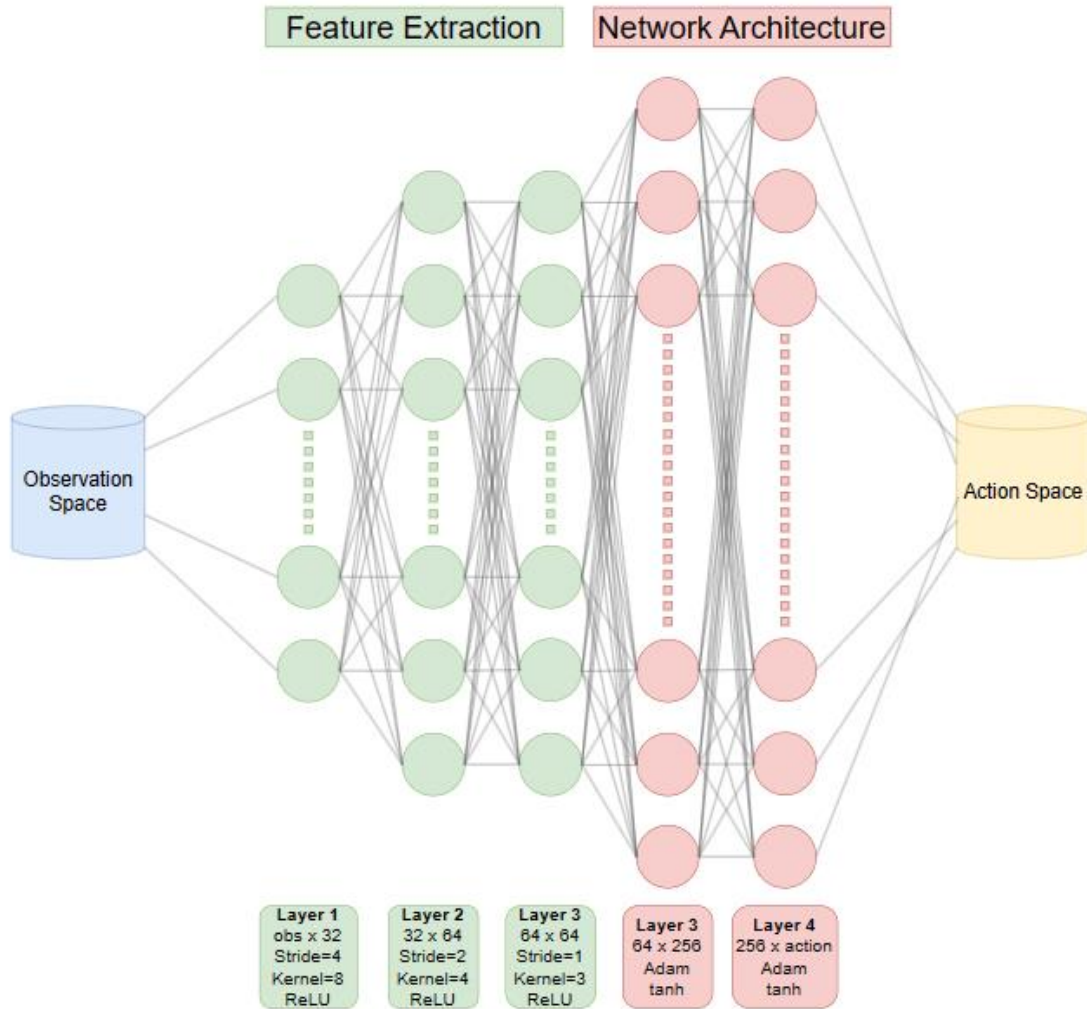


Figure 4-24 TQC Policy Network

4.4. Concluding Remarks

In this section, the data architecture designed to train and test deep reinforcement learning algorithms on hydraulic models to optimise pressure management was explained thoroughly. This included various sections that explained the leakage problem further and redefined it in terms of DRL formalisms; details of designing and building the central GYM compatible environment and a brief description of the main agents utilised in the case studies.

Initially, the differences between leakage types (background and bursts) were highlighted followed by the main objectives of the leakage problem. Pressure management is used as a contingency for background leakage and a temporary solution for burst events. Hydraulic

models are created by integrating various data sources such as geographical information, network data, operational data, and customer data. Hydraulic solvers are used for modelling WDNs and execute steady state and extended period simulations. Leakage events are introduced in the model using pressure dependent equations and coefficients to control the magnitude. The leakage pressure relationship is manipulated to identify the emitter coefficient ranges that describe background and burst leakage in literature. It was found that emitter coefficients between 0 to 0.196 signify background leakage and 0.196 onwards are burst leakage. Furthermore, the leakage problem is contextualised with the concepts of Markov Decision Process (MDP) highlighting key MDP components like rewards, actions, states, and environments. It describes how pressure management in WDNs can be approached as a reinforcement learning (RL) problem within the MDP framework. RL terms are also defined with respect to the context of WDN pressure management. Insights into the challenges of managing leakage in water distribution networks help set the stage for developing a reinforcement learning-based solution within the defined MDP framework. The WDN-DRL ecosystem is fully realised in the schematic displayed in **figure 4-3**.

In more detail, section 4.2 provides a comprehensive overview of the design and components of the RL environment, tailored for addressing the leakage problem in water distribution networks. The design and components of the reinforcement learning (RL) environment in the leakage problem is explained. The environment should be challenging but not excessively difficult. It must provide observable states and an appropriate action space to allow the agent to make informed decisions. The environment leverages the hydraulic capabilities of EPANET using a python wrapper (EPYNET) to enable programmatic interaction. Incremental steady-state simulations are employed to simulate the effects of actions and leakage throughout the day. OpenAI Gym is used to define action and observation spaces, providing extensibility and compatibility with RL libraries like Stable Baselines 3, TensorFlow and PyTorch. Various space types, including Box, Discrete, and Multi-Discrete, are explained and selected based on the problem's nature to realise the environment. Subsequently, the fundamental functions of the environment are clarified. The step function uses the actions chosen by the agent to interact with the hydraulic model to update the environment state, calculate the rewards, and log data. Formulating the reward is crucial to guide the agent's behaviour and reflect how well it accomplishes objectives. The reward function is derived to evaluate penalties for leakage events before and after an action has been executed. Scale parameters in the reward function are tuned to balance objectives. Additionally, agent-environment interactions are visualised through five bespoke renderings. These figures use logged data to aid with debugging, analysis

and fine-tuning DRL algorithms. All the data produced from interactions in the environment can also be logged to produce step and episode reports.

Finally, DRL agents are selected based on the availability of algorithms compatible with the box action space. The highest performing algorithms used in the case studies are discussed in detail. The choice of agents and their neural network architectures is based on their performance in the specific problem and the ability to capture long-term dependencies and patterns in the data. Further tuning of hyperparameters are conducted to improve agent training and performance.

5. Background Leakage Case Study

Unlike burst events, background leakage is difficult to detect hence it is necessary to create pressure controllers that are sensitive to these small changes and account for the water loss they incur. In this chapter we detail the methodology developed to introduce background leakage events to two water distribution networks (benchmark and real) and use a variety of DRL and non-DRL optimisation algorithms to minimise the adverse effects of the leakage. The reward function is optimised to ensure the objectives are achieved. This is followed with diagrams and graphs explaining the results of the various algorithms and their ability to optimise the reward and minimise water loss. Finally, the results are discussed thoroughly, and conclusions are drawn. The main aims of this experiment are:

- Provide a method to mitigate background leaks in real-time through pressure control.
- Test the viability and scalability of different DRL models in WDN pressure management.
- Highlight the differences between DRL performances to popular optimisation algorithms.
- Discuss the performances of different DRL methods and the effects of hyperparameter tuning.

The full case study including python files, figures, excel reports and hydraulic files can be accessed privately on GitHub by following this [link](#). In addition, the main results and figures for this chapter is included in Appendix F and G.

5.1. Methodology

5.1.1. Optimisation algorithms

The non-DRL algorithms used includes a Nelder Mead (NM) algorithms to benchmark numerical optimisation methods for non-differential objective functions. NM tends to produce satisfactory results with low computational effort. The Nelder Mead algorithm is created with a maximum number of function evaluations of 1000, absolute acceptable error in inputs between iterations is 0.005, and absolute acceptable error in output between iterations is 0.01. A Particle Swarm optimisation (PSO) algorithm is developed using 30 particles and 20 generations in each run to increase the search space and reach a global optimum. It is based on the idea of swarm intelligence. The use of PSO is recorded in optimisation of WDN pressure management in (Mehdi and Asghar, 2019). Hence why, PSO was used as a benchmark for meta-heuristic search algorithms. Differential evolution (DE) was chosen as the benchmark

evolutionary algorithm due its ability to outperform general genetic algorithms (GA). We use a DE algorithm with a population size of 30, 20 generations, a crossover rate of 0.25 and scale factor of 1. It is another meta-heuristic algorithm with wide application in water distribution based on the idea of selective breeding and evolution (Hajgató, Paál and Gyires-Tóth, 2020; Bilal and Pant, 2022). Whilst meta-heuristic approaches do not guarantee a global optimum, they tend perform quite well in non-differentiable applications. The code used to build the benchmark non-DRL algorithm is written in the Appendix B for reference.

DRL optimisation included the gradient based policy algorithms Trust Region Policy Optimisation (TRPO), Proximal Policy Optimisation (PPO) and Recurrent Proximal Policy Optimisation (Recurrent PPO). Hybrid value-driven and policy driven algorithms such as Advantage Actor Critic (A2C), Soft Actor Critic (SAC), Deep Deterministic Policy Gradient (DDPG), are also used. An additional RL policy method that utilises random search of the policy parameter is used for its ability to compete with the more sophisticated DRL algorithm which is named the Augmented Random Search (ARS) algorithm. Finally, a distributional DRL algorithm is also used which is called Truncated Mixture of Continuous Distributional Quantile Critics (TQC). This cohort of DRL algorithms will be initially tested using their basic hyperparameters before tuning the highest three performers.

5.1.2. Problem setup

Appropriately, the problem is initiated as a network with small leaks on every node. This leak is calculated through the pressure-dependent leakage rate described in **equation 2-9**. The leakage coefficient K_f is redefined depending on the discharge coefficient c and the length between nodes i and j (L_{ij}) as described in **equation 3-2** and kept under the background coefficient limit of $0.196 \text{ L s}^{-1} \text{ m}^{-0.5}$ as calculated in section 4.1.1. Following that, the networks will utilise a universal leakage exponent of 1.18 which is the widely accepted value in literature (Araujo *et al.*, 2006; Saldarriaga and Salcedo, 2015b). The leakage rate and violation count at each node before and after PRV action will be evaluated to explore the effect of the action on the network. Algorithm 5-1 outlines the pseudo code of the background leakage case study.

Algorithm 5-1: Background leakage scenario pseudo code

Input: link flows and nodal pressures

Output: Reward

for each episode **do**

 initialise all valve settings to be 40.

 get link flow and nodal pressures for leaking network.

 get leakage rate of leaking network.

 initialise state s .

for each step of episode, state s is not terminal, timestep is not 24.

do

$a \leftarrow$ action given by optimisation algorithm for state s .

 evaluate network after action reward.

 get link flow and nodal pressures for solved network.

 get new leakage rate of solved network.

 reward r given by difference in leakage rate and pressure violations.

 laziness penalty if settings haven't changes in three steps.

 get new state s' .

 log rewards, violations, leakage.

 take action a , observation, r , s' .

$s \leftarrow s'$

end

end

In this scenario, we imagine that the network is inherently filled with the undetected background leakage forming our 'leaking network' followed with an action to then create the 'solved network' as displayed in **figure 5-1** below. The reward (r) function will be calculated based on equation 3-4 that calculates the difference between both networks' weighted penalties. The reward scales are modified to navigate the objective trade-off between water loss and pressure violations iteratively using the method described in section 4.2.4. for each experiment.

The limitations of this study includes its reliance on clear trusted data and optimised valve locations. We also require reliability evaluations before deploying these algorithms on water networks. Another limitation is the unknown effects of the optimisation algorithms on other

objectives of the WDN. In essence, focusing on pressure control for leakage management might affect other WDN objectives in asset preservation.

We have therefore assumed that the hydraulic models can serve the digital twin of the real WDNs and that the pressure-leakage relationship is defined accurately by **equation 2-9**. More assumptions include accurate carbon emissions conversion figures as recorded in (Department for Energy Security and Net Zero, 2023).

More details on the limitations and assumptions made in this study can be found in sections 7.1 and 7.2.

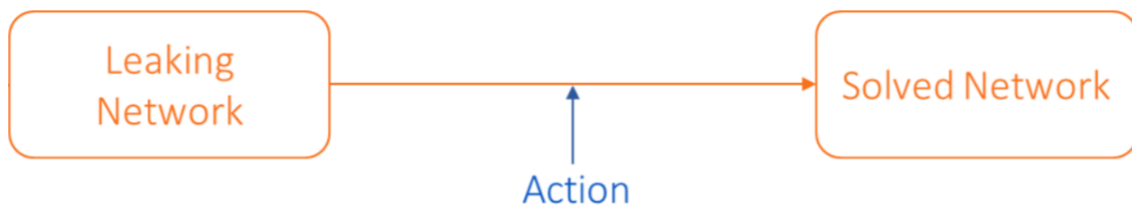


Figure 5-1 Background leakage scenario flow

5.1.3. Testing

In the testing stage, all optimisation algorithms were subjected to three full training episodes consisting of 24 timesteps each modelling an hour of the customers demand patterns. The resulting rewards and leakage were recorded and plotted for further data analysis and comparisons. The time required to optimise pressure management was recorded to highlight the computational effort incurred by each algorithm. Deep RL algorithms require training before they are applied to the environment hence the separate training and test time; whilst the other optimisation algorithms can be deployed instantly to the environment. These experiments, also consider the computational effort required to optimise the scenarios. This is displayed in a bar chart where the DRL test times are plotted separately from the training times. After the entire cohort of DRL algorithms are tested and plotted against the benchmark optimisation algorithms, the highest performing DRL algorithms are selected for further hyperparameter tuning and re-tested. This is to highlight the effect of hyperparameters on the training and performance of DRL algorithms. To facilitate that, a new evaluation method is developed to create and test several algorithms under a parameter sweep. In **figure 5-2** below, the wall plot of the learning rate and discount factor sweeps of the advantage actor critic algorithm is shown. Learning rate is a parameter between 0 and 1 which is measure of exploration in the DRL algorithms where higher exploration is 1 and high exploitation (no exploration) is a 0. On the other hand, the discount factor is a measure of foresight the

algorithm should have. A discount factor (ranging from 0-1) of 0 does not consider any future consequences to the current action and only focuses on the current reward whereas a value of 1 forces the algorithm to consider all the consequent future rewards. The figure highlights the effect of changing the parameter on the training curve of the DRL algorithm and how tuning it can improve performance.

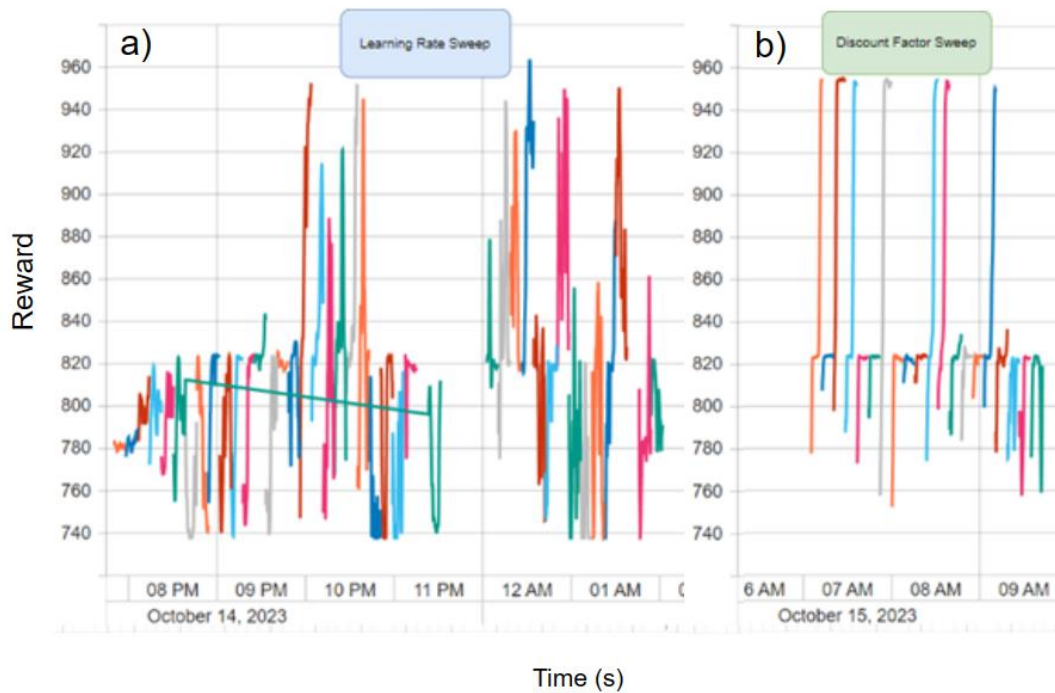


Figure 5-2 Hyperparameter Sweep. a) Learning Rate b) Discount Factor.

Whilst the sweep proved the importance of hyperparameters; the model’s sensitivity to hyperparameters varied. Generally, these sweeps helped advise which values should be avoided in algorithm design and which are favourable. Additional experiments tested functionality such as linear scheduling of the learning rate and different combinations of the discount factors.

5.2. Jowitt & Xu Network

The case study selected for this research is the medium-sized benchmark test network for most pressure management applications in WDN; the Jowitt & Xu network. This network was first introduced in (Jowitt and Xu, 1990) and quickly became the research community’s standardised benchmark. The network consists of 22 nodes, 37 pipes and 3 tanks. Due to the extensive work on choosing the correct valve location in previous research, we adopt three valve locations presented in (Araujo *et al.*, 2006). **Figure 5-3** below illustrates the network structure. The valve locations are denoted by the two opposite facing triangles whilst tanks are shown as the half-filled rectangles and nodes are represented as dots. Finally, the nodes are

connected through a series of pipes which are shown as lines. The EPANET simulation helps provide a colour map of current nodal pressures and link flows.

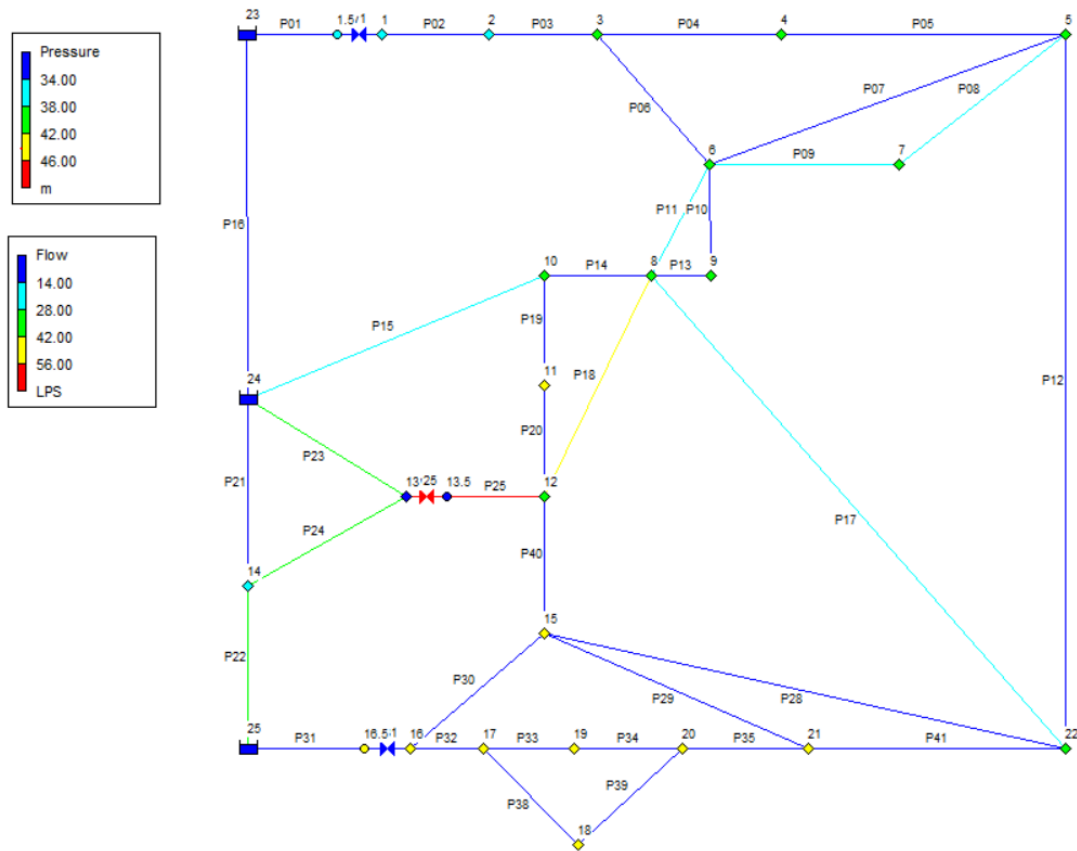


Figure 5-3 Labelled Jowitt & Xu network including tanks, PRVs, nodes, and pipes.

The pipe properties, nodal emitter coefficients and demand patterns were all modelled after the standard test network (Araujo *et al.*, 2006, p. 138,139). The network properties and data are all displayed below in **Tables 5-1** and **5-2**.

Table 5-1 Benchmark pipe and node data

Pipe ID	Length (m)	Diameter (mm)	Roughness ($m^{1/3}s^{-1}$)	Node ID	Elevation (m)	Base Demand (Ls^{-1})	Emitter ($Ls^{-1}m^{1/2}$)
P01	606	457	110	Junc 1	18	5	0.012055
P02	1930	457	110	Junc 2	18	10	0.033656
P03	5150	305	10	Junc 3	14	0	0.032088
P04	326	152	100	Junc 4	12	5	0.005562
P05	844	229	110	Junc 5	14	30	0.018383
P06	1274	152	100	Junc 6	15	10	0.019238
P07	1115	229	90	Junc 7	14.5	0	0.0053
P08	500	381	110	Junc 8	14	20	0.018853
P09	615	381	110	Junc 9	14	0	0.003532
P10	300	229	90	Junc 10	15	5	0.019837
P11	743	381	110	Junc 11	12	10	0.00627
P12	1408	152	100	Junc 12	15	0	0.02441
P13	443	229	90	Junc 13	23	0	0.016842
P14	249	305	105	Junc 14	20	5	0.01949
P15	3382	305	100	Junc 15	8	20	0.028884
P16	454	457	110	Junc 16	10	0	0.013467
P17	931	229	125	Junc 17	7	0	0.010957
P18	1600	457	110	Junc 18	8	5	0.005286
P19	542	229	90	Junc 19	10	5	0.009203
P20	777	229	90	Junc 20	7	0	0.010819
P21	2782	229	105	Junc 21	10	0	0.020118
P22	304	381	135	Junc 22	15	20	0.034997
P23	1767	475	110				
P24	1014	381	135				
P25	762	457	110				
P28	2334	229	100				
P29	832	152	90				
P30	914	229	125				
P31	1097	381	6				
P32	822	305	140				
P33	1072	229	135				
P34	864	152	90				
P35	711	152	90				
P38	411	152	100				
P39	701	229	110				
P40	1996	229	95				
P41	2689	152	100				

Table 5-2 Benchmark consumption factors data and reservoir levels

Time (s)	1	2	3	4	5	6	7	8	9	10	11	12
Fc	0.61	0.61	0.41	0.41	0.41	0.41	0.81	0.81	1.23	1.23	1.13	1.13
23	55.2	55.3	55.5	55.6	55.7	55.8	55.9	56	55.7	55.4	55.2	55.1
24	55.2	55.3	55.3	55.4	55.4	55.5	55.5	55.5	55.3	55.2	55	54.8
25	55	55.1	55.2	55.3	55.4	55.4	55.5	55.5	55.5	55	54.8	54.7
Time (s)	13	14	15	16	17	18	19	20	21	22	23	24
Fc	0.92	0.92	0.92	0.92	1.03	1.03	0.92	0.92	0.82	0.82	0.61	0.61
23	54.9	54.7	54.6	54.6	54.5	54.5	54.6	54.7	54.8	54.9	55	55.2
24	54.8	54.8	54.7	54.6	54.6	54.5	54.7	54.7	54.7	54.8	54.9	55
25	54.5	54.4	54.3	54.1	54	54	54.2	54.3	54.5	54.6	54.8	54.9

The PRVs were installed to the benchmark on pipes P31, P01, and P25 using invisible nodes Junc 16.5, Junc 13.5, and Junc 1.5 that don't affect the hydraulic simulation. These locations were chosen from literature (Araujo *et al.*, 2006) as the best valve locations for the pressure management of the Jowitt & Xu test network. Finally, the reward formulation was completed through testing different reward scale ratios as detailed in section 4.2.4. The results proved that a ratio of 3:1 favouring the leakage objective produces the best trade-off between the two objectives of leakage and pressure violation minimisation. The pressure limits placed on this water network encourage that nodal pressures remain between a minimum of 10m and a maximum of 70m.

5.2.1. Results

When the algorithms were tested to minimise the background leakage, most of them managed to solve the problem effectively and optimise the reward. However, their performance varied greatly. The algorithms' rewards over the three-day period were recorded in boxplot form with the mean reward marked with an 'x' and the inner points marked with an 'o' in **figure 5-4**. Box plots help demonstrate the algorithm's performance throughout the episodes for clear comparisons. Smaller box plots highlight the algorithm's ability to achieve reproducible results. In the initial testing stage, the difference in performance between the DRL models and the benchmarked optimisation algorithms was clear as seen in **figure 5-4**. The differential evolution (DE) algorithm scored the highest in reward maximisation at 41.61. This is followed by the PSO at 40.95 and NM algorithm at 40.19. The stability of the DE algorithm was a clear indication that the best result has been achieved by DE. The highest DRL algorithm was the augmented random search (ARS) which displayed 13% decrease in performance in comparison to DE. This was followed by the distributional DRL algorithm truncate quantile critics (TQC) and its actor-critic predecessor soft actor critic (SAC). The main portion of the DRL models (PPO,

Recurrent PPO, A2C, TRPO) performed similarly with an average reward of 30-35 apart from DDPG that yielded the worst performance with an average reward of 22.37.

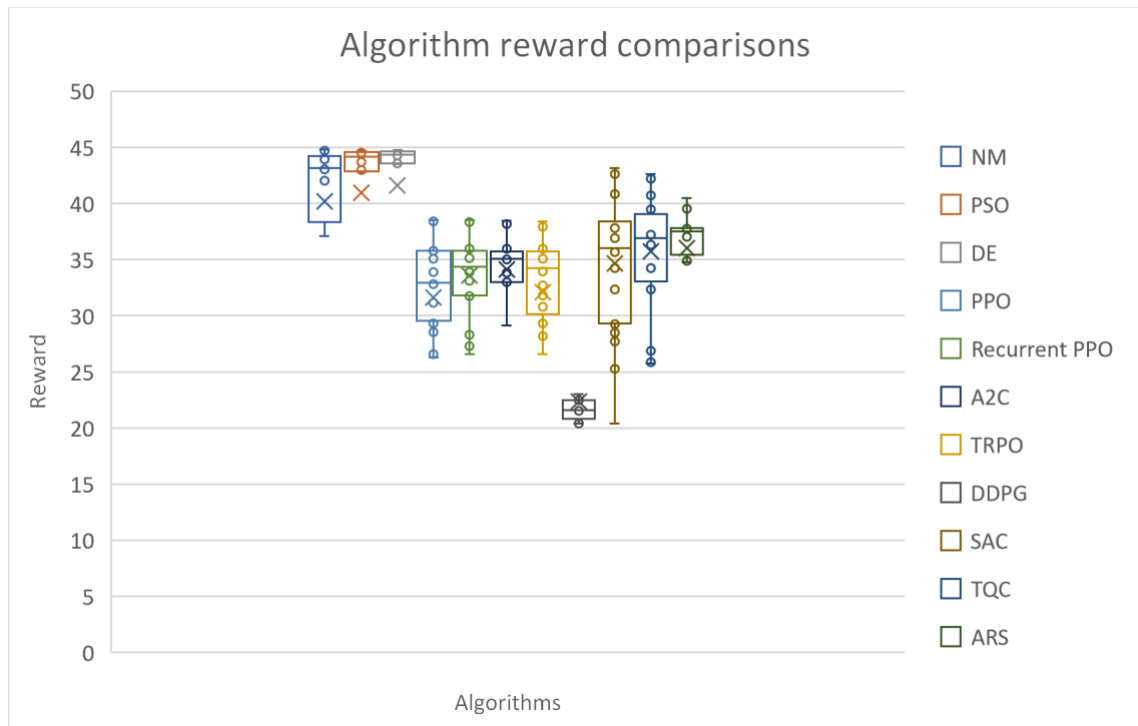


Figure 5-4 Initial algorithm performance - Jowitt & Xu network

The corresponding processing speed is displayed in **figure 5-5**. Non-DRL computational time is recorded directly whilst DRL algorithms' computational time is split into training time and test time. Training time denotes the time required to build and train the DRL model on the environment before it is deployed into the testing environment. All DRL methods were subject to an identical 20,000 timesteps of training to develop their neural network weighting and minimise loss. Testing time consists of the time required to process all 72 timesteps that make up the case study, saving logged data, and creating data visualisation figures. All training and testing runs are performed using an AMD Ryzen 4700U, 200MHz, 8 processor CPU. The DRL models far surpassed the benchmark models as displayed in **figure 5-5**. Processing times for DRL algorithms are split into two sections: The training time to build the DRL model; and the test time to run the background leakage case study. Hence, the test time represents how fast the trained models would interact with the network and is equivalent to the benchmark's running time. The benchmark algorithms solved the case studies at various speeds with the fastest being the Nelder-Mead optimisation algorithm (NM) needing 233.3s to complete the problem, followed by the highest performing algorithm DE at 676s and PSO which required 1274s. In comparison, the DRL algorithms increased computational efficiency and speed resulting in the fastest DRL model to complete the leakage problem in 22.1s which was the

DDPG model. The rest of the DRL models performed in the range of 22-30s with the slowest being the TQC model at 30.1s. This signifies an increase of processing speed equivalent to 7.8-10.6x between the DRL algorithms and the Nelder-Mead model and a boost of 42.4-57.4x with the slowest benchmark algorithm (PSO). In addition, the DRL algorithms' computational speeds corresponds to a range of 0.3-0.4s per timestep which more than qualifies the method to real-time control.

Within the field of DRL algorithms, all training loops were limited to 20,000 timesteps for fair comparison. This resulted in widely varying training times due to the methodology required to train the models. The slowest model to develop was the TQC algorithm which required 1312.9s to train followed by the SAC at 797.4s. The rest of the DRL models required training times varying from the fastest (PPO) at 216s and 401s for the recurrent PPO model. Essentially, training times do not affect the algorithms' real time performance but produces insight on the computational loads of building the DRL models.

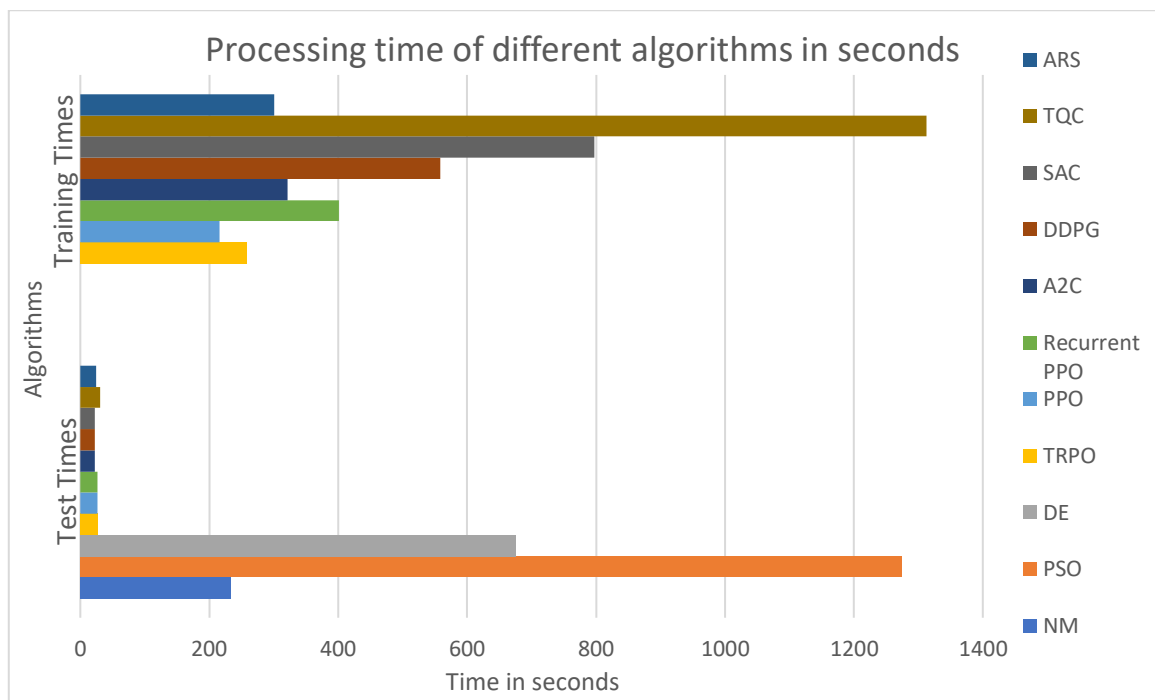


Figure 5-5 Initial algorithm speed - Jowitt & Xu network.

In the initial simulations the DRL models were trained using the original parameters set by Stable Baselines 3 libraries. These parameters, explained in **table 5-3** below, were tuned using the sweep function shown in **figure 5-2** where multiple agents were trained using a sweep function on the key parameters to highlight the best values for DRL model training.

Table 5-3 DRL algorithm training hyperparameters.

Name	Policy network	Timesteps	Learning rate	Discount factor	Value coefficient	Clip range
PPO	MLP	20,000	0.0003	0.99	0.5	0.2
Recurrent PPO	MLP-LSTM	20,000	0.0003	0.99	0.5	0.2
A2C	MLP	20,000	0.0007	0.99	0.5	NA
TRPO	MLP	20,000	0.001	0.99	NA	NA
DDPG	MLP	20,000	0.001	0.99	NA	NA
SAC	MLP	20,000	0.00077	0.9	0.5	NA
TQC	MLP	20,000	0.0003	0.99	NA	NA
ARS	MLP	20,000	0.02	NA	NA	NA
Name	Policy network	Timesteps	Learning rate	Discount factor	Soft update coefficient	Buffer size
SAC-Tuned	MLP	20,000	Scheduled 0.001	0.55	0.01	1e6
TQC-Tuned	MLP	30,000	Scheduled 0.0015	0.25	0.005	1e6
Name	Policy Network	Timesteps	Learning rate	Evaluating episodes	Exploration noise	Random perturbations
ARS-Tuned	Linear	30,000	0.02	5	0.05	8

The highest DRL performers of the initial simulation were SAC, TQC and ARS which were customised through hyperparameter tuning to produce better results. Designing these algorithm's parameters through tuning and research produced new results which were displayed in **figure 5-6**. The tuned DRL algorithms are contrasted to the original settings and the benchmarked optimisation algorithm to highlight the effect of hyperparameter tuning and the DRL performance with respect to the benchmarked algorithms. Furthermore, the DRL algorithms' theory is explained in **section 4.3** and their parameters are explained in detail in **Appendix C**.

The results showed a visible increase in performance for all three DRL algorithms pushing them within 5.5% from the highest performer (DE). The new rankings place the TQC and SAC models on par with the second place PSO algorithm at 40.59 and 40.72 average rewards respectively.

This is followed by the former best DRL algorithm ARS which now yields 39.3 average rewards just under the benchmark NM algorithm. This leap in performance closes the gap between the evolutionary algorithm DE and the DRL algorithm SAC where SAC has a 2.15% decrease in performance and a 25x boost in speed as shown in **figure 5-7**.

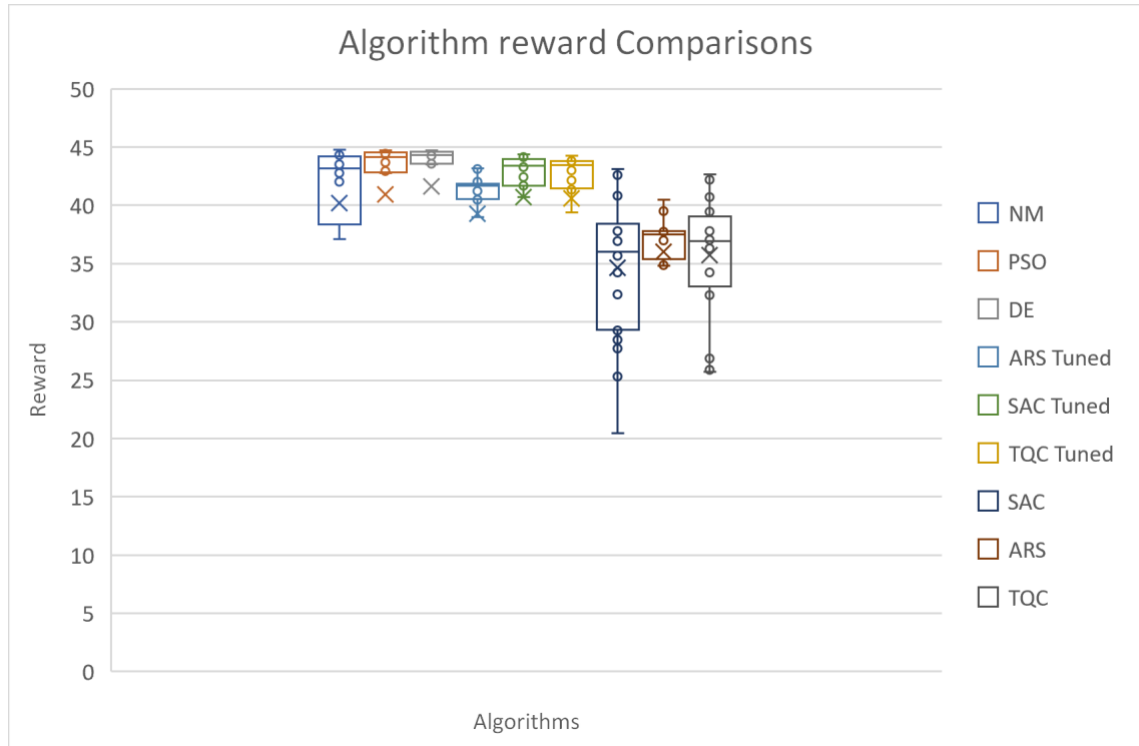


Figure 5-6 Tuned algorithm performance - Jowitt & Xu network.

In addition, the new processing time of the tuned DRL algorithms are plotted in **figure 5-7** to provide insight on computational efficiency. Similar to **figure 5-5**, the processing time of the tuned DRL algorithms is split into training time (bottom) which is the time taken to develop and build the DRL models through a training loop (top) which is the time taken for the DRL algorithms to run the 3-day test scenario. The advantage of processing speed remains with the tuned TQC and ARS algorithms. Hyperparameter tuning has therefore unveiled a best new trade-off through SAC's performance and speed making it the third best performing algorithm and third fastest value for processing speed as shown in **table 5-4**.

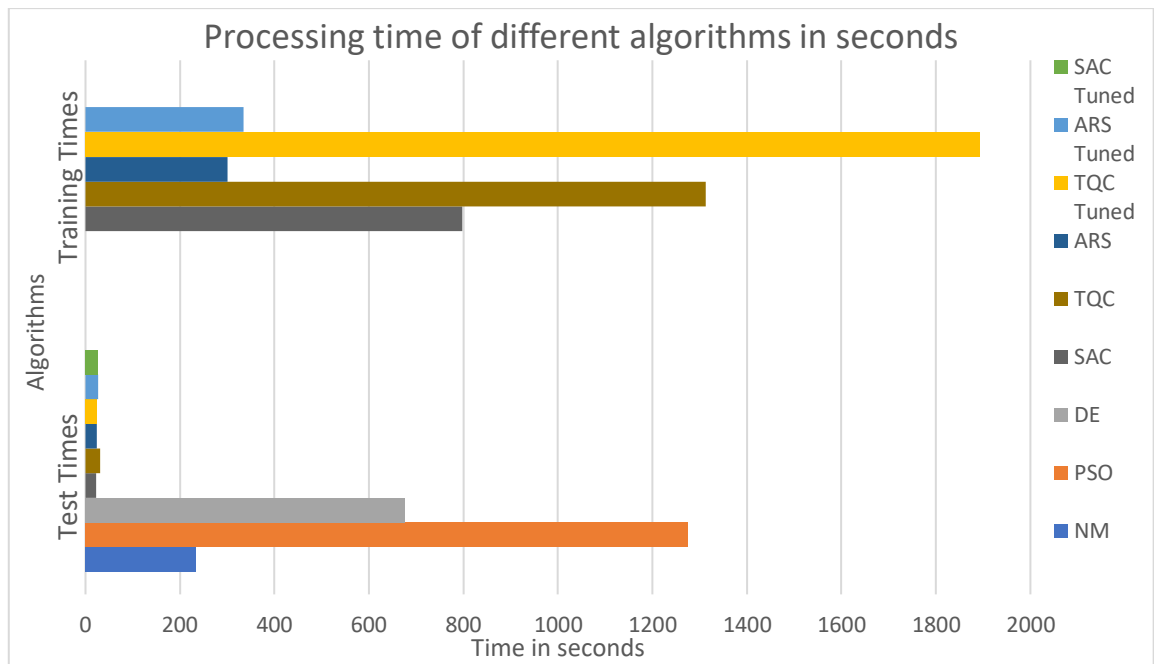


Figure 5-7 Tuned algorithm speed - Jowitt & Xu network.

Moreover, the key episodic results of the case study are displayed below in **table 5-4**. The algorithms' performances in maximising rewards, maximising water saved from leakage, minimising pressure violations, maximising carbon emission reductions and processing time. The carbon emissions are calculated using water leakage measurements and the conversion factor (176.7 KgCO₂/million litres) dictated by the UK's government values for the transport and distribution for 2023 presented in (Department for Energy Security and Net Zero, 2023). The best result in each category is marked in bold. PPO and Recurrent PPO models yielded the highest % of water saved at 73.4% and the highest carbon emissions reduction with 302.5 kgCO₂ reduced. However, this came at the expense of pressure violations that accumulated to the values of 300 for Recurrent PPO and 304 for PPO. In contrast, DE showed the best pressure violations minimisation with 46 violations and the best average episodic reward 995.2. Finally, the best processing speed was achieved by PPO for training time and DDPG for test time. **Table 5-4** shows how deploying the optimisation algorithms can lead to significant reductions in leakage and pressure violations. We discuss these results further and draw conclusions in the next section.

Table 5-4 Key results – Jowitt & Xu

Algorithm	Average Reward	Average Water Saved (%)	Average Pressure Violations	Carbon Emissions Reduction (KgCO2)	Training Time (s)	Test Time (s)
NM	965.7	66.0	49	273.2	NA	233
PSO	994.5	65.9	48	272.8	NA	1274
DE	995.2	65.9	46	269.2	NA	676
ARS-Tuned	943.5	65.5	78	270.9	335	26.6
SAC-Tuned	975.0	64.2	49	269.5	850	27
TQC-Tuned	976.3	65.2	50	270.1	1892.3	24.8
TRPO	800.5	73.2	264	301.7	259	27
PPO	780.0	73.4	304	302.5	216	26.6
Recurrent PPO	798.4	73.4	300	302.5	401	27
DDPG	525.0	44.1	48	181.1	558	22.1
A2C	822.5	73.2	300	301.8	321.5	22.7

5.2.2. Discussions

In this section, we discuss and evaluate the results of the Jowitt & Xu network for the background leakage case study presented in section 5.2.1. The main objectives of this case study was to minimise leakage through pressure valve control and minimising pressure violations for the benchmarked test network presented in (Jowitt and Xu, 1990).

Initial Simulation

When challenged with controlling PRVs in the Jowitt & Xu network, all optimisation algorithms performed favourably however results varied greatly. The DRL algorithm’s lower performance can be explained by their untuned parameters, yet they remain the most attractive option due to their high computational efficiency. All untuned DRL algorithms performed less favourably than the three benchmark optimisation algorithms in the initial simulation. Nevertheless, the

DRL algorithms have shown their ability to tackle complexity through their deep neural network's function approximation capabilities. This was reflected in the processing times displayed in **figure 5-5**. DRL processing times described both computational needs to develop a policy (training time) and computational loads to solve the case study (test time). The test time demonstrated the computational efficiency of the DRL models in solving the case study with respect to the benchmark algorithms. The best DRL algorithm in terms of reward accumulation through this simulation (ARS) produced a 27.5x increase in computational speed to the best benchmark optimisation algorithm (DE).

Furthermore, the training times highlight the DRL models' ability to develop a beneficial policy using 20,000 timesteps of data. Shorter training times spotlights algorithms that can be easily modified and retrained as the network changed for continuous deployment. This could manifest as a continuous improvement (CI) loop that can adjust DRL models to seasonality trends throughout the year. Comparing the speed and performance results, unveils an interesting trade-off where the faster DRL models produce weaker performances to the slower benchmarks. The ARS, TQC and SAC models lie in the middle of the trade-off producing both high performances and a great increase in computational efficiency. Hence why, they were selected for hyperparameter tuning through a sweep function.

Tuned Simulation

Tuned DRL models were built for the top 3 performing DRL algorithms and contrasted to their original performances and the benchmark optimisation algorithms. Tuning the DRL hyperparameters increased the DRL models' capabilities to reach higher rewards and even ranking TQC in the top 3 overall performances. This improvement seen through tuning with a simple sweep function provides an insight to how hyperparameter optimisation through expert systems such as Optuna can affect DRL capabilities (Akiba *et al.*, 2019). For the purposes of this case study, the tuned algorithms have shifted the trade-off between performance and computational efficiency further to their favour. With the distributional DRL algorithm (TQC) performing within 5% of the current best practice of DE and providing a 25x increase in speed.

This allows DRL algorithms to act in real-time to solve the pressure management problem whilst evolutionary algorithms such as DE are usually paired with model predictive control (MPC) (Sadler *et al.*, 2020). The current standard of MPC often requires the use of high-performance computing facilities to run simulations based on less accurate forecast data. In essence, deploying a DRL algorithm through MLOps (Machine Learning Operations) and a continuous improvement – continuous deployment architecture (CI/CD) can provide water

utilities a method to pressure control DMA-based networks in real time. This will further increase leakage minimisation as new settings can be assigned at intervals as small as 1s providing capabilities to fine-tune the pressure management policy. The current practice is pressure valves being controlled hourly.

DRL Models Comparisons

The deep deterministic policy gradient (DDPG) belongs to the hybrid methods used in this experiment. It exploits both value-driven, and policy-driven neural networks to converge at an optimised solution. However, due to its sensitivity to hyperparameters and tendency to converge at suboptimal policies, DDPG found itself performing worst in the DRL cohort. This could also be explained through DDPG's use of deterministic policies which are less effective when dealing with temporal events such as demand pattern-influenced hydraulics. On the other hand, its high convergence properties materialised in test time making it the fastest model to solve the case study even if it was not well optimised. In comparison, Advantage Actor Critic (A2C) developed a better policy with both higher water saved % and lower pressure violations. Even though both algorithms belong to the same category of hybrid methods, A2C largely outperformed DDPG which could be due to its stochastic policy that incorporates exploration directly to its policy. The final hybrid method was SAC which is an off-policy algorithm that improves on its predecessor (actor critic) through entropy regularisation. This allows it to navigate the exploration-exploitation trade-off even better by inherently including entropy in its value function. This simple change has increased the performance to the second highest DRL algorithm with an episodic reward of 975. It is clear that better-suited algorithms can navigate the exploration-exploitation trade off more effectively and deal with the stochasticity of the environment. This is also represented during SAC tuning where changing the discount factor (γ) has improved performance greatly. This was observed in all three tuned algorithms. SAC results showed a better compromise between leakage minimisation and pressure violations. It managed to save more water than DDPG at 64.2% and less pressure violations than A2C at 49 violations making it the best hybrid method in the DRL cohort.

Building on SAC's performance, the truncated quantile critics (TQC) algorithm alters the value function slightly to produce a value distribution over the states rather than an absolute sum. This qualifies it as distributional DRL techniques and explains the name as it truncates the critic network's value quantiles. This distribution managed to improve performance slightly making TQC the highest performing DRL algorithm at 976 episodic rewards. This translated into a slight increase in water saved and 50 pressure violations when compared to its predecessor SAC.

In retrospect, policy driven methods (TRPO, PPO, Recurrent PPO, ARS) couldn't match their hybrid alternatives. This can be explained with the policy-driven methods' lower stability and sensitivity to changes in the environment due to the absence of the value function. TRPO tackles this instability through trust region methods. However, the trust region constraints led to conservative policy updates and hindered the algorithm's ability to explore better solutions. This affected the algorithm's performance making it less beneficial than hybrid methods yet an improvement on other policy-driven methods (PPO, Recurrent PPO) with 800.5 episodic rewards. The rewards were manifested through a great performance in leakage minimisation (73.2%) at the expense of pressure violations (264). Comparable performances were achieved by PPO and Recurrent PPO agents that exploited the leakage objective at the expense of the pressure violation objective both saving the highest amount of water at 73.4% and violating 304 and 300 pressure limits respectively. Their performances in leakage minimisation have also translated in carbon emissions reduction placing them joint first with episodic reductions of 302.5kg of CO₂. Recurrent PPO's ability to minimise pressure violations better placed it ahead of its predecessor PPO. This could be due to the long short-term memory (LSTM) functionality placed on the neural networks however their results are too close to draw a valid conclusion. Finally, the most effective policy driven agent was the augmented random search (ARS) algorithm. ARS managed to provide a better trade-off between saving water and pressure regulation by focusing more on pressure regulation than its neighbours PPO and recurrent PPO. As a result, it developed a policy that produced lower violations (78) at the expense of lower water saved (65.5%). ARS learns through random search on the parameterised policy equation making it more sample efficient but slightly less stable. Nevertheless, its simple exploration strategy has waged well in the case study.

5.3. Northumbrian Water Network

In contrast to the Jowitt & Xu network, the second case study tackles a much larger system provided from a real network under the supervision of Northumbrian Water Living (NWL). NWL has agreed to the use of their hydraulic data and models to simulate the control of pressure reducing valves (PRVs) and the throttle control valves (TCVs) for the purpose of pressure management and leakage reduction. Whilst the Jowitt & Xu network represents the size of approximately 1-2 DMAs in what is considered a medium sized WDN problem; NWL's SZ08 model represents a full water distribution system consisting of 19 DMAS and records of 36 unique demand patterns for residential building, police stations, hospitals, commerce and more. The complexity of this case study increases exponentially when comparing network sizes hence increasing computational load and testing the limits of the optimisation algorithms

tested. A summary of network parameters of SZ08 and Jowitt & Xu is shown below in **table 5-5**.

Table 5-5 Summary of network parameters

Name	Junctions	Pipes	Valves	Pumps	Tanks/Reservoirs	DMAs	Size
Jowitt & Xu	25	37	3	0	3	1	Medium
SZ08	1988	2022	32	26	11	19	Extra Large

Controlling the entirety of SZ08 should serve as a challenging task due to complexity alongside the predetermined locations of valves. SZ08's valves have been placed many years ago to control the inflow to DMAs and tanks amongst other objectives whilst the Jowitt & Xu valve locations were based on optimised locations for pressure regulation and leakage management derived from literature. Hence, valve control is expected to be less effective in NWL's case study. In essence, the primary goal of this case study to experiment with DRL's scalability and performance under high complexity. Due to the size of the network, it is difficult to provide a fully annotated figure. Therefore, the network architecture and key visualisations will be displayed in **figure 5-8** and **figure 5-9**.

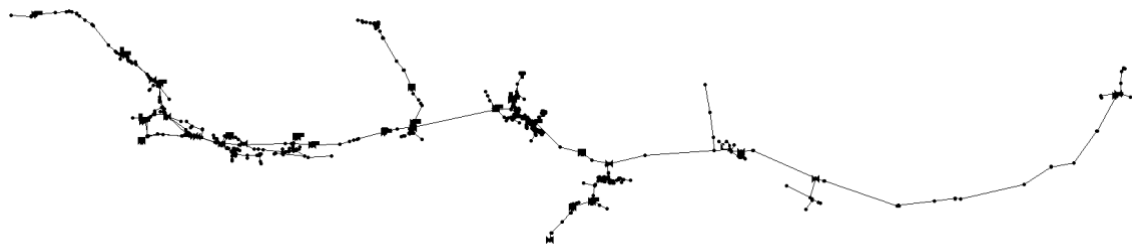


Figure 5-8 SZ08 network architecture

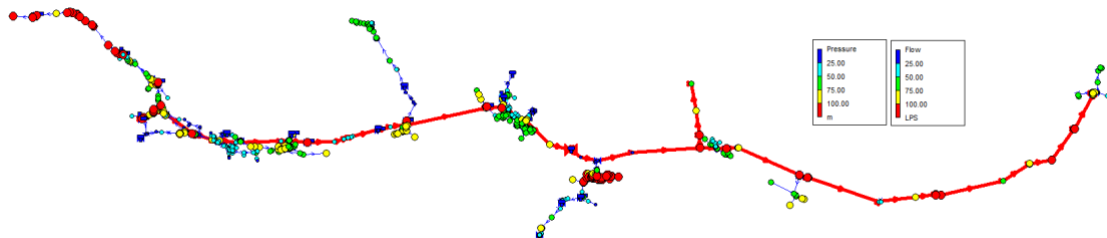


Figure 5-9 SZ08 key visualisation of high pipe flows and nodal pressures.

In order to initialise the experiment, emitter coefficients had to be introduced to the SZ08 model to simulate background leakage. This is achieved through manipulating **equations 2-9**

and **3-2** to find the appropriate emitter coefficient within the background leakage emitter range of 0 to $0.196 \text{ Ls}^{-1}\text{m}^{-0.5}$. Hence, emitter coefficients were assigned based on the neighbouring pipe lengths where the node with the largest neighbouring pipe length was assigned an emitter coefficient of 0.196. The remaining nodes' emitter coefficients were calculated as a ratio to the maximum pipe length value as described in **equation 5-1**.

$$k = \frac{\sum_j^M L_{ij}}{\max(L)} \times 0.196 \quad (5-1)$$

Where k is the emitter coefficient ($\text{Ls}^{-1}\text{m}^{-0.5}$), L_{ij} is the pipe length between nodes i and j (m), M is the total number of nodes and $\max(L)$ is the maximum connected pipe length. Using this equation, we can ensure that nodal emitter coefficients remain proportional to the length of connected pipes as described in **equation 3-2** and within the calculated limit of background leakage (0.196).

Furthermore, an initial simulation was conducted to choose the best reward scales for this particular case study as describe in section 4.2.4. The results unveiled that a ratio of 3:1 favouring leakage management over pressure violations produces the best trained DRL algorithms for the objectives at hand. This is demonstrated in **figure 5-10** below which highlights how agents with a 3:1 ratio minimised penalty best. The pressure limits of this experiments encourage nodal pressures to be kept between 10m and 200m. The higher upper limit of 200m is placed to allow for the high pressures observed in trunk mains.

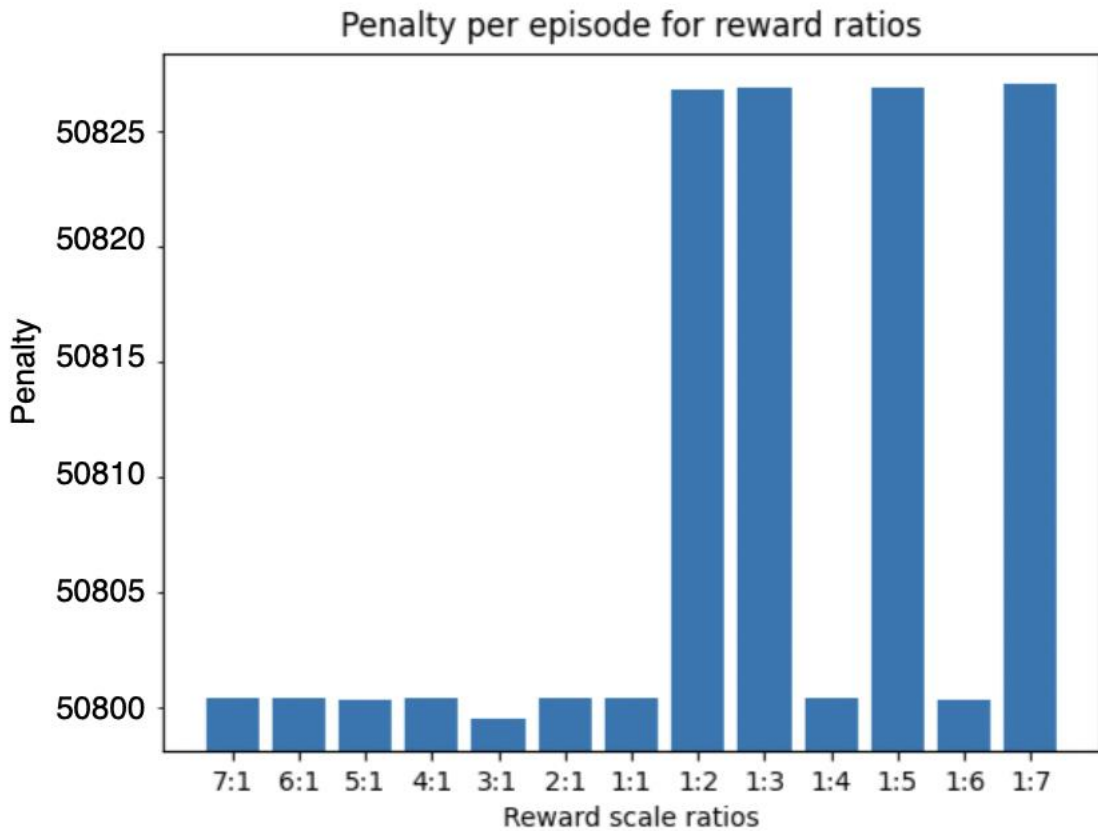


Figure 5-10 Episodic penalty plots for different reward scales - SZ08

The benchmark optimisation algorithms (NM, PSO, DE) and DRL algorithms (PPO, Recurrent PPO, TRPO, A2C, SAC, TQC, ARS, DDPG) are tested using the same network and environment using their initial algorithms to evaluate performance and speed. In this case study the DRL algorithms were not subjected to any hyperparameter tuning due to the high computational load of training agents. The hyperparameters used for the DRL algorithms are listed below in table 5-6.

Table 5-6 DRL agent hyperparameters - SZ08

Name	Policy network	Timesteps	Learning rate	Discount factor	Value coefficient	Clip range
PPO	MLP	20,000	0.0003	0.99	0.5	0.2
Recurrent PPO	MLP-LSTM	20,000	0.0003	0.99	0.5	0.2
A2C	MLP	20,000	0.0007	0.99	0.5	NA
TRPO	MLP	20,000	0.001	0.99	NA	NA
DDPG	MLP	20,000	0.001	0.99	NA	NA
ARS	MLP	20,000	0.02	NA	NA	NA

Name	Policy network	Timesteps	Learning rate	Discount factor	Soft update coefficient	Buffer size
SAC	MLP	20,000	0.00077	0.9	0.005	1e6
TQC	MLP	20,000	0.0003	0.99	0.005	1e6

5.3.1. Results

Tackling the SZ08 case study proved more challenging for all optimisation algorithms as they attempted to fulfil the objectives of leakage minimisation and pressure regulation. This was due to multiple factors that include the increased complexity through larger observation and action spaces along with the sub-optimal valve locations. Nevertheless, the optimisation algorithms succeeded in minimising leakage and pressure violations collecting positive rewards. The positive rewards signify the model's improvement on the current settings of the networks which would gain a reward of zero. The reward spread across the network junctions are plotted using the data visualisation tools mentioned in chapter three and displayed in **figure 5-11**.

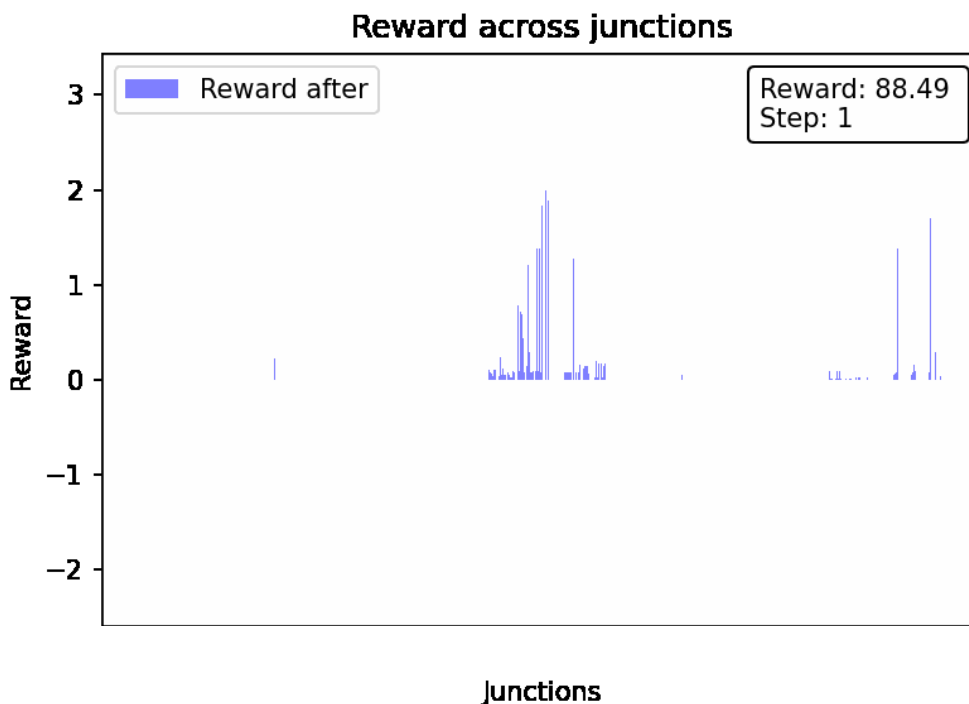


Figure 5-11 Reward spread across junctions.

When evaluating the rewards distribution across the junctions in the network, all the different valve settings were only able to enact hydraulic change over nodes in the middle and right side

of the plot. The unaffected junctions display an area of the network that is not covered hydraulically by the current collection of pressure valves. Using this plot, we could identify areas in the network that are more vulnerable to background leakage as they cannot mitigated by pressure management.

Rewards collected were gathered in a box plot for each algorithm to highlight their performance and draw comparisons in **figure 5-12**. In this figure, rewards during each step of the case study were plotted using a 'o' and the average was represented with an 'x' across the box plot. The algorithms generally performed comparably with the benchmark optimisation methods gaining a slight advantage on the DRL models. Benchmark algorithms scored the highest performances with DE achieving 89.74, PSO at 89.04, and NM at 87.11. The DRL models were led with TRPO which managed to exploit a beneficial trust region and producing average rewards at 82.36 followed by a drop to SAC which lies at the 78.52 mark. Most DRL models performed within the 75-80 reward range except the DDPG model which incurred the worst result at 32.19.

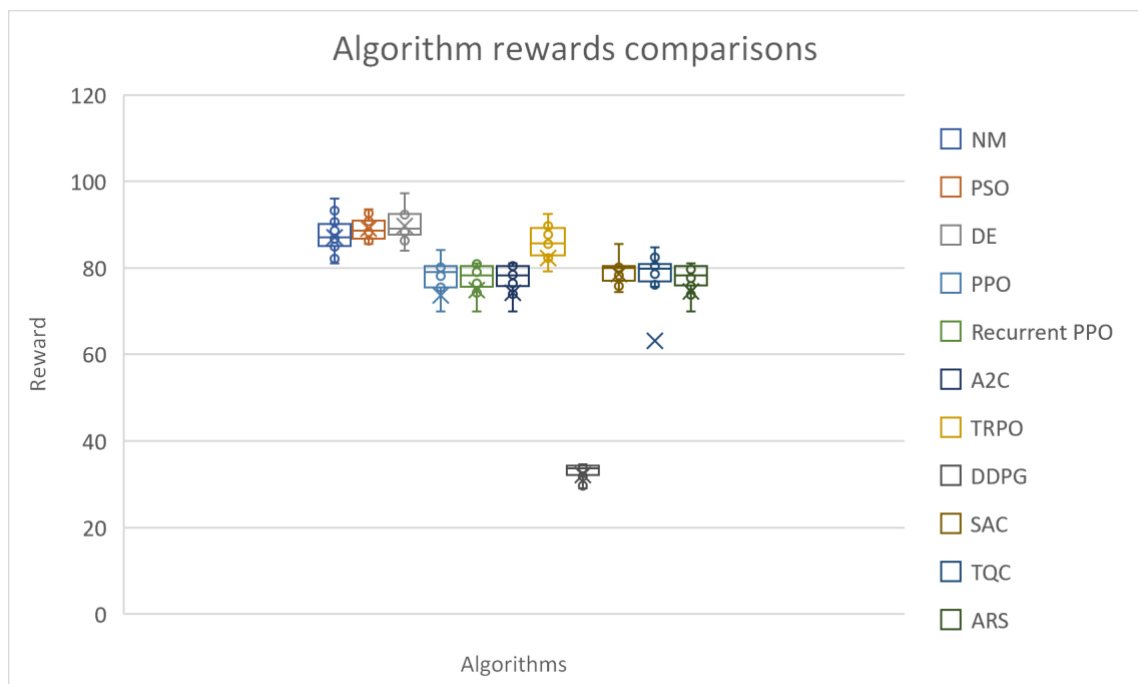


Figure 5-12 Algorithm performance - SZ08

The processing speed associated with each method was plotted in a bar graph displayed in **figure 5-12**. The optimisation algorithms were graded based on processing time for benchmark algorithms and test and training times for DRL algorithms. Training times for DRL models denotes the period required for the algorithms to be developed through 20,000 iterative

timesteps whilst the test time denotes the time required to solve the case study using observations. Therefore, the DRL test times and the benchmark processing times signify the algorithms' computational efficiency in solving the SZ08 case study.

Generally, the DRL models have managed to decrease processing time considerably in comparison to the benchmark algorithms providing an 8.39x speedup in computational time of the highest performing DRL model (TRPO) in comparison to the highest performing benchmark algorithm (DE). The Nelder-Mead algorithm produced the slowest processing time demanding 23238s which equates to 6 hours, 27 minutes and 18 seconds followed by PSO at 12909.4s and DE at 7013.1s. In terms of test time, also known as implementation time, DRL models produced faster processing speeds. The fastest model was ARS needing 691.6s and the slowest being TRPO taking 835.7s to solve the SZ08 network.

All DRL models were assessed on their training time with the longest trained algorithm being the TQC followed by its non-distributional predecessor SAC at 6760s and 6422s respectively. The fastest training simulation was achieved by the TRPO at 2667s and PPO at 3535s. Training time could be beneficial as it indicates the computational expenses required to create and update the DRL models.

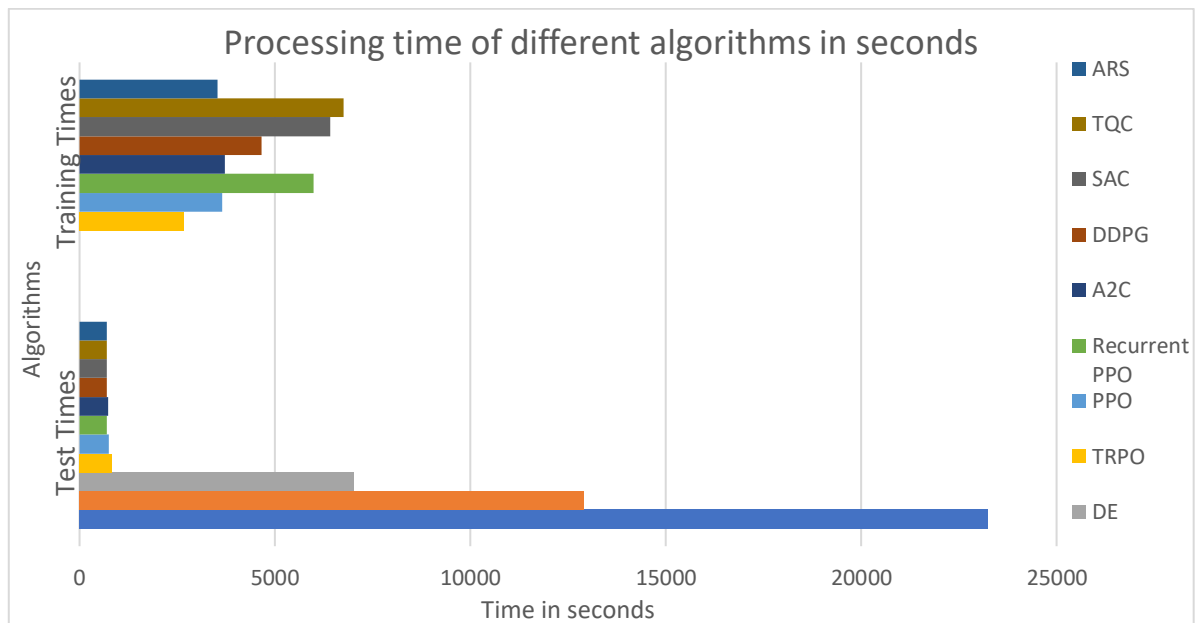


Figure 5-13 Algorithm speed - SZ08

Overall, a summary of the key episodic results of the experiment is collected in **table 5-7**. The algorithms are evaluated based on rewards, water saved, pressure violations, carbon emissions, and processing time. The best result for each category is also marked in bold to insert an additional perspective to the algorithms' performances. Differential evolution

optimisation managed to strike the best trade-off between pressure violations and water saved, hence gathering the highest average episodic reward. Jointly, the PSO and NM algorithms minimised average episodic pressure violations with 2284 violations. The best performance in minimising leakage produced a meagre 0.693% water savings which was awarded to the TRPO algorithm. The 0.693% water saving corresponds to the highest savings in episodic carbon emissions of 169.3 kg CO₂ saved. Finally, the TRPO also produced the best training time amongst DRL models at 2667s whilst ARS solved the testing scenarios under 692s.

Table 5-7 Key results - SZ08

Algorithm	Average Reward	Average Water Saved (%)	Average Pressure Violations	Carbon Emissions Reduction (KgCO2)	Training Time (s)	Test Time (s)
NM	2097	0.626	2284	156.4	NA	23238
PSO	2140	0.633	2284	159.8	NA	12909.4
DE	2153	0.646	2286	160.3	NA	7013.1
ARS	1775	0.690	12604	169.0	3535	691.6
SAC	1852	0.679	12608	166.2	6422	704.9
TQC	1606	0.667	12584	163.3	6760	699.5
TRPO	1892	0.693	22955	169.8	2667	835.7
PPO	1686	0.691	12604	169.2	3652	741.9
Recurrent PPO	1768	0.691	12604	169.2	5989	700.8
DDPG	772.6	0.268	12276	66.02	4656	695.7
A2C	1773	0.691	12604	172.7	3726	740.3

5.3.2. Discussions

The discussion section details insights on the system zone 08 network (SZ08) case study provided by Northumbrian Water Living (NWL) which models the water distribution network for Lumley, England. The experiment initiates appropriate emitter coefficients to model background leakage and minimises its detrimental effects through pressure management. By controlling pressure reducing valves (PRVs) and throttle controlling valves, optimisation algorithms can attempt to minimise water lost through leakage and nodal pressure violations. The SZ08 network tests the scalability of DRL models through its high complexity and multiple dimensions. Results can be summarised in terms of performance (reward collection) and speed (running time).

Performance

SZ08 tested the algorithms' abilities to navigate high dimensional problems due to the network's size and differing components. Unlike the previous experiment, DRL models were not hyperparameter tuned as the process would be too computationally expensive given the complexity of the case study despite its guaranteed improvement in policy formulation and performance. The untuned DRL models were trained and tested on the SZ08 against three

benchmark optimisation models (Nelder Mead, Particle Swarm Optimisation, and differential evolution). Most algorithms managed to solve the case study effectively apart from DDPG that converged at a sub optimal result due to its deterministic nature. Despite the heightened ability to minimise pressure violations, the tested optimisation algorithms were unable to minimise leakage past 0.693%. This was a downgrade in comparison to the previous Jowitt & Xu network however could be explained by the lower valve to node ratio and inefficient valve locations. Overall, the benchmark DE algorithm performed the best as it collected 2153 episodic rewards by striking the best balance between leakage minimisation and pressure violations although the NM and PSO algorithms received the joint best result in minimising pressure violations. The PSO algorithm scored second (2140) place due to its performance in pressure management followed by NM which scored third (2097) due to its lower capabilities in leakage minimisation.

Surprisingly, the DRL cohort was led by the TRPO model which proved to be more effective in high-dimensional problems. Its trust region exploitation has equipped it with the stability required to solve this case study. At a mere 12% decrease in performance from the DE algorithm, TRPO received 1852 episodic rewards and the highest overall water saved (0.693%) This led to the highest carbon emissions reduction with 169.8kg of CO₂ saved. The other DRL models in the policy driven family scored considerably lower with ARS at 1775 episodic rewards followed by the Recurrent PPO and PPO methods at 1768 and 1686 respectively. These results highlight the importance of stability to tackle the SZ08 problem with more stable DRL algorithms scoring higher. In addition, Recurrent PPO's ability to perform higher than the PPO model proves the importance of neural network architecture in optimisation as the major difference between the two models is the inclusion of the long short-term memory (LSTM) module in the Recurrent PPO neural network architecture. LSTM cells can store the sequentially processed data to represent long-term dependencies which make it an attractive choice in time-related problems.

The hybrid DRL methods were led by SAC which was the second highest performing DRL model after TRPO with 1852 episodic rewards and lower pressure violations of 12604 rather than TRPO's 22955 violations. With the exception of the advantage actor critic (A2C) model, the remainder of the hybrid DRL models performed poorly. A2C saved 0.691% of the water and landed on 1773 episodic rewards which places it in between ARS and Recurrent PPO in terms of performance. However, TQC and DDPG algorithms had the lowest performances with 1606 and 773 episodic rewards respectively further proving the importance of developing a stochastic policy rather than a deterministic policy in DDPG. TQC's performance leads us to

believe that the distributional DRL methods do not tackle complexity well due to the large state space that forms a scarce value probability distribution.

Speed

A large benefit of using DRL for optimisation purposes is its ability to build intelligence and minimise computational load. Hence why, comparing DRL model's speed to the benchmark algorithms is necessary. **Figure 5-12** displays the associated processing time for all the optimisation algorithms including training and test times for the DRL cohort. All algorithms were tasked with completing 3 episodes consisting of 24 steps each to model three 24-h days. The processor used was an AMD Ryzen 4700U, 200MHz, 8 processor CPU. In contrast to the Jowitt & Xu case study, the SZ08 problem incurred much heavier computational loads resulting in slower processing speeds. This was apparent in NM's processing which lasted 23,238 seconds which is equivalent to approximately 6 hours and 27 minutes making it the slowest algorithm. As expected, the other benchmark algorithms were slower than all the DRL models with PSO requiring 12909 seconds and DE with a significant improvement at 7013s.

When evaluating DRL algorithms, it was necessary to differentiate between training time used to develop a policy and test time used to implement the policy. Training time provides insight into algorithm design and possible computational expense for retraining whilst test time provides insight into whether they can be viable to real-time control or rather the standard model predictive control. TRPO held the fastest training time only demanding 2667s to develop the best performing DRL policy although it required 835.7s to solve the problem making it the slowest test time. ARS produced the second fastest training time with 3535s and the fastest test time with 691.6s. Therefore, the fastest DRL test time (ARS: 691.6s) produces a 10x improvement in computational speed than the fastest benchmark algorithm (DE: 7013s). This is due to ARS' low computational load since it utilises a simple random search method over the parameterised policy equation. The rest of the DRL cohort incurred a training time ranging between 1-2 hours with the slowest being TQC demanding 6760s of training. Algorithms with large neural networks such as TQC and SAC often demand long training times as displayed in both experiments.

5.4. Concluding Remarks

Chapter 5 included two centrepiece experiments that illustrated DRL algorithm's ability to minimise background leakage through advanced pressure managements. Valve control of a benchmark test network (Jowitt & Xu network) and a real water network (SZ08) provides the variety needed to highlight the scalability and computational efficiency of DRL techniques.

Initially, the methodology used for the background leakage case studies was detailed. The benchmark optimisation algorithms were used to contextualise the DRL cohorts and draw comparisons with the current best practices. The experiment develops a differential evolution algorithm in addition to the particle swarm optimisation and the Nelder Mead algorithm. The problem setup dictated the process used to design the experiment through a foundation of equations and algorithms derived from literature. The test scenarios used to evaluate algorithm performances and the metrics recorded were designed to highlight scalability, computational demand, and the effects of hyperparameter tuning. Furthermore, this chapter introduced the novel use of eight DRL algorithms for advanced pressure management as viable real-time alternative optimisation algorithms. The optimisation objectives were to simultaneously minimise nodal pressure violations and background leakage within the network through valve control. The DRL agents developed were tested alongside benchmark optimisation algorithms to assess their performance and computational efficiency.

The Jowitt & Xu network was the benchmark used to experiment the effects of pressure management on minimising background leakage. The DRL cohort consisting of 8 algorithms is compared to the benchmark algorithms with a focus on their performance and speed. The results are displayed in figures and a summary table highlighting key metrics such as carbon emissions, water saved, reward collected and more. The best performing DRL algorithms were tuned by changing the hyperparameters to demonstrate how DRL algorithms can be improved further through hyperparameter optimisation.

SZ08 water network was collected and developed by Northumbrian Water to control the DMAs based in Lumley, England. This network is inherent with more complexities due to size, various customer demands, and network components as shown in **table 5-5** and **figures 5-8 & 5-9**. Background leakage is introduced to the nodes and the optimisation algorithms mentioned above are used to solve the test scenarios. The results are compared through an extensive discussion of algorithm performance and speed.

Whilst the DRL algorithms on average performed less favourably than the current best practice; several models managed to perform comparably to the DE and PSO algorithms with little to no hyperparameter tuning. Hyperparameter tuning and training methods have had a great impact on algorithm performance making it clear that further improvement is possible through hyperparameter optimisation. Whilst hybrid methods were more effective in developing DRL models for the smaller case study, it lacked the stability offered by policy driven methods for the larger application. Considering their significant increase in

implementation time and the lower computational load, DRL algorithms provide a true promise for real-time control (less than 0.4s per prediction) on a DMA level. However, due to the complexity of whole water distribution systems shown in SZ08, real-time control is slightly out of the scope of DRL using regular processors (around 10s per prediction). Using high end computing could improve that greatly and make real-time control of whole system zones a reality. The use of real-time control in WDN pressure management is a novelty that is bound to increase water savings and carbon reductions as they allow valves to react more instantly to changes in the network whereas the current standard receives signals hourly or even daily.

Comparing the two case studies, it is clear that the optimisation approach is more beneficial on a DMA scale however still applicable on a large scale. The SZ08 case study provides a more complex large-scale problem with multiple demand patterns and varying topographies. Therefore, DRL algorithms that prioritise stability and sample efficiency such as TRPO perform better. On the other hand, Jowitt & Xu provides a smaller scale and better valve locations resulting in better overall results. DRL algorithms with better exploration properties managed to overcome local optima found at 780-800 with high water saved% and high-pressure violations. Thus, hybrid methods such as A2C and SAC performed well whilst the distributional DRL method TQC performed best. The difference in performance between the two case studies could suggest that optimising on a DMA scale could affect neighbouring areas resulting in additional trade-offs. DRL models can be trained to enact pressure control policies that significantly minimise water loss on the DMA level nevertheless future work should focus on experimenting with multi-agent DRL control. Using this technology, it may be possible to train agents individually on DMAs and have them communicate through an overarching MADRL system. Hence, this ensures that we harness the possible savings on a DMA level without creating sub-optimal states on neighbouring DMAs in a more open-looped fashion. The design of DRL algorithms is very complex and is brim with possibilities for customisation hence future work should encourage the use of different neural network architectures, optimisers, activation functions and pre-training techniques.

6. Burst Leakage Case Study

The contrast between burst and background leakage lies within the size of the leak. The definition of burst leaks are leakage events that are detectable through modern detection techniques hence they are often repaired quickly. Therefore, burst leakage cause less damage than background events on the long run. Burst events are represented in literature by $0.5\text{m}^3/\text{h}$ flow rates at a pressure of 50m (García and Cabrera, 2007). This is modelled by an emitter coefficient greater than $0.196\text{ L s}^{-1}\text{m}^{-0.5}$ as derived in chapter 4.1.1. Throughout the burst leakage case study, we will detail the methodology used to introduce burst events to the water distribution network at random nodes followed by pressure valve commands. The pressure management commands will be fed from one of three benchmark non-DRL algorithms or the eight DRL algorithms being tested. The reward function is formulated through comparisons of DRL algorithms trained using different scale ratios. After that, the algorithms solve the benchmark Jowitt & Xu network and the real SZ08 hydraulic model. The results for both networks are displayed through various box plots, bar graphs and summary tables. Finally, the findings are discussed, and conclusions are drawn. The main aims of this case study are the following:

- Provide a method to mitigate random bursts in real-time through pressure control.
- Test the viability and scalability of different DRL models in WDN pressure management under randomised burst conditions.
- Highlight the differences between DRL performances to popular optimisation algorithms.
- Highlight differences and similarities between the performances of the DRL agents in burst leakage reduction.

The full case study including python files, figures, excel reports and hydraulic files can be accessed privately on GitHub by following this [link](#). In addition, the main results and figures for this chapter is included in Appendix H and I.

6.1. Methodology

6.1.1. Problem Setup

Burst leakage is easily detectable however they can cause huge water loss, and, in most cases, there are no real-time contingencies to minimise the leakage other than pipe repair/replacement. Thus, we have devised this scenario to test the effect of pressure management on mitigating leakage through burst events. In this scenario we evaluate the

distribution of the water network in question with randomised burst events carrying a leakage discharge coefficient larger than 0.196 forming the 'leaking network'. This is meant to model irreversible pipe failures with high water loss. Bursts count will match the number of pressure valves in the network to make sure that both networks have the same capabilities at solving the problem. The randomised burst events will force the optimisation algorithms to use the observation data to make connections between pressure fluctuations, leakage and PRV locations. Therefore, when a sudden change in pressure appears in the observation data, the optimisation algorithms are expected to identify that as a leak implicitly and moderate the closest valve to minimise leakage. In the interest of fairness, all optimisation algorithms will be developed from an environment with randomised bursts and applied to the test scenarios.

Following that, the networks will utilise a universal leakage exponent of 1.18, which is the widely accepted value in literature (Araujo *et al.*, 2006; Saldarriaga and Salcedo, 2015b), to complete the leakage rate equation. In response to the leakage, our optimised agent (i.e., valves guided by the optimisation algorithm) will produce an action to mitigate this leakage, hence creating the 'solved network'. The leakage rate and violation count at each node before and after PRV action will be evaluated to explore the effect of the action on the network. The difference in performance between the solved and leaking networks will prove as a measure of the effect of the action on mitigating the leak. **Algorithm 6-1** below describes how the burst leakage scenario is performed.

Algorithm 6-1 Burst scenario pseudo code

Input: link flows and nodal pressures

Output: Reward

for each episode **do**

 initialise all valve settings to be 40.

for each step of episode, state s is not terminal, timestep is not 24.

do

 introduce leakage nodes and magnitude.

 evaluate leaking network penalty.

$a \leftarrow$ action given by optimisation algorithm for state s .

 evaluate network after action penalty.

 reward r given by difference in leakage rate and pressure violations.

 laziness penalty if settings haven't changes in three steps.

 get the new state s' .

 log reward, violations, leakage metrics

 take action a , observation r, s' .

$s \leftarrow s'$

end

end

The reward (r) function will be calculated based on **equation 3-4** that calculates the difference between the 'solved' and 'leaking' networks' weighted penalties. The reward scales are modified to navigate the objective trade-off between water loss and pressure violations iteratively using the method described in section 4.2.4. for each network.

6.1.2. Testing

As the environment is run in episodes consisting of 24 timesteps, each timesteps draws from customer demands to model the hydraulic requirements for every hour of the day. Hence why, the testing stage will consist of the optimisation algorithms solving three episodes of data and the rewards, leakage and violations will be reported. In addition, the computational effort required will be represented as the time needed to complete the test cases. Whilst the non-DRL algorithms can tackle the problem in situ, the DRL algorithms must be trained to develop their policies and adjust their deep neural networks weightings. The developed DRL model will be tested on the same scenarios as the benchmark models. The aim of this case study is to highlight the ability of DRL algorithms to effectively manage pressure valves under randomised burst locations. The algorithms will be judged on their ability to minimise pressure violations

and leakage control in the WDN. The different networks are used to test the scalability of the DRL models in medium and large complex problems.

6.2. Jowitt & Xu Network

The Jowitt & Xu network is the name given to a standardised benchmark network that is popular in the research community. It was first introduced in (Jowitt and Xu, 1990) and consists of 22 nodes, 37 pipes and 3 tanks. The WDN architecture is presented in **figure 6-1** with a pressure and flow colourmap to help understand the state of the network. Many researchers produced results explaining the optimal valve locations for pressure management in Jowitt & Xu however the most effective arrangement was mentioned in (Araujo *et al.*, 2006). Adopting the same number of valves and locations from (Araujo *et al.*, 2006), PRVs are placed on pipes P01, P31 and P25 also shown in **figure 6-1**.

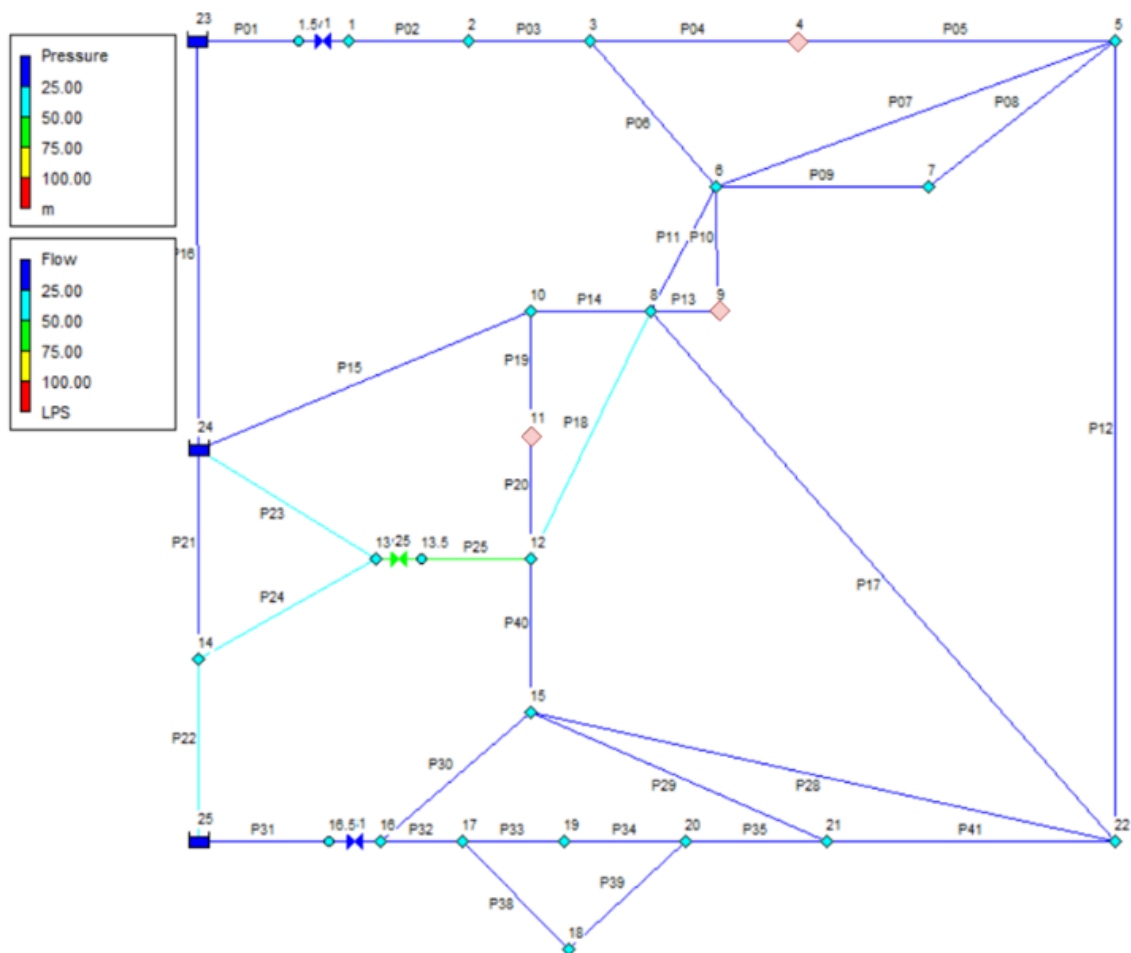


Figure 6-1 Labeled Jowitt & Xu network including bursts (red) tanks, PRVs, nodes, and pipes.

The testing network will include three randomly located bursts introduced with an emitter coefficient of 3 whilst the other nodes' emitter coefficients are initialised at 0 to extenuate the effects of bursts on water distribution. The randomiser has selected nodes '4', '9' and '11' as

the burst locations marked in red in **figure 6-1**. The network hydraulic properties described above are also detailed in **table 6-1** with the slight amendment to emitter coefficients. The customer demand patterns are identical to the ones used in **table 5-2** to keep the benchmark identical.

Table 6-1 Benchmark pipe and amended node data.

Pipe ID	Length (m)	Diameter (mm)	Roughness ($m^{1/3}s^{-1}$)	Node ID	Elevation (m)	Base Demand (Ls^{-1})	Emitter ($Ls^{-1}m^{-1/2}$)
P01	606	457	110	Junc 1	18	5	0
P02	1930	457	110	Junc 2	18	10	0
P03	5150	305	10	Junc 3	14	0	0
P04	326	152	100	Junc 4	12	5	3
P05	844	229	110	Junc 5	14	30	0
P06	1274	152	100	Junc 6	15	10	0
P07	1115	229	90	Junc 7	14.5	0	0
P08	500	381	110	Junc 8	14	20	0
P09	615	381	110	Junc 9	14	0	3
P10	300	229	90	Junc 10	15	5	0
P11	743	381	110	Junc 11	12	10	3
P12	1408	152	100	Junc 12	15	0	0
P13	443	229	90	Junc 13	23	0	0
P14	249	305	105	Junc 14	20	5	0
P15	3382	305	100	Junc 15	8	20	0
P16	454	457	110	Junc 16	10	0	0
P17	931	229	125	Junc 17	7	0	0
P18	1600	457	110	Junc 18	8	5	0
P19	542	229	90	Junc 19	10	5	0
P20	777	229	90	Junc 20	7	0	0
P21	2782	229	105	Junc 21	10	0	0
P22	304	381	135	Junc 22	15	20	0
P23	1767	475	110				
P24	1014	381	135				
P25	762	457	110				
P28	2334	229	100				
P29	832	152	90				
P30	914	229	125				
P31	1097	381	6				
P32	822	305	140				
P33	1072	229	135				
P34	864	152	90				
P35	711	152	90				
P38	411	152	100				
P39	701	229	110				
P40	1996	229	95				
P41	2689	152	100				

Finally, the reward scales for DRL were also selected through training a DRL algorithm using different scales. The resultant DRL models were tested to evaluate their ability to optimise the primary objectives (leakage and pressure violations). Comparing the penalties incurred by

these models in **figure 6-2** unveils the best trade-off being at scaling the objectives at a ratio of 1:1. Due to the randomised nature of the leaks, the reward scales were tested several times to ensure that this ratio is optimal which it was.

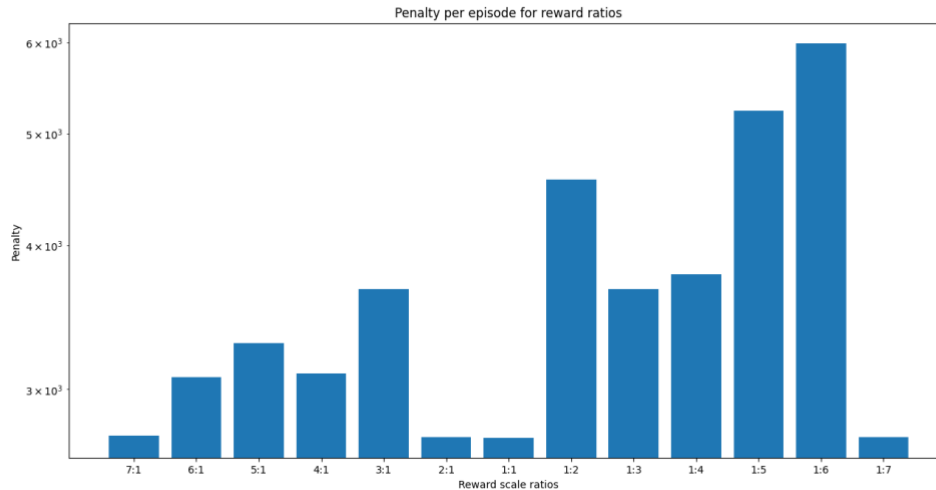


Figure 6-2 Episodic penalty for reward ratios

6.2.1. Results

The case study aims to focus on two main aspects of the results which are performance and speed. The performance results can be represented by the rewards gained throughout the testing episodes displayed in **figure 6-3**. In this box plot, individual step rewards gained at each hour are represented by an 'o' and the mean step reward is marked with an 'x' while the box marks the interquartile range. Tackling randomised bursts is an issue that requires machine intelligence due to the added complexity. Hence why, there is a clear improvement in performance when deploying DRL models over the benchmark optimisation algorithms. DRL models learnt to flag changes in the observation space (nodal pressure readings) caused by leakage and react by managing the nearest pressure valve. The best performance overall was achieved by the Advantage Actor Critic (A2C) algorithms with an average of 1.76 step reward followed closely by ARS (1.73) and Recurrent PPO (1.71). Their smaller interquartile ranges highlight their ability to adapt to the varying demands of the episode (daily customer demands) by modifying the valve settings whilst algorithms that exploit safe beneficial settings such as TRPO show a larger interquartile range. The benchmark algorithms, led by the particle swarm optimisation algorithm, have collected the lowest rewards due to their inability to rely on experience to influence their actions. To improve their results, they need to rely on highly accurate predictive models and are insufficient on their own.

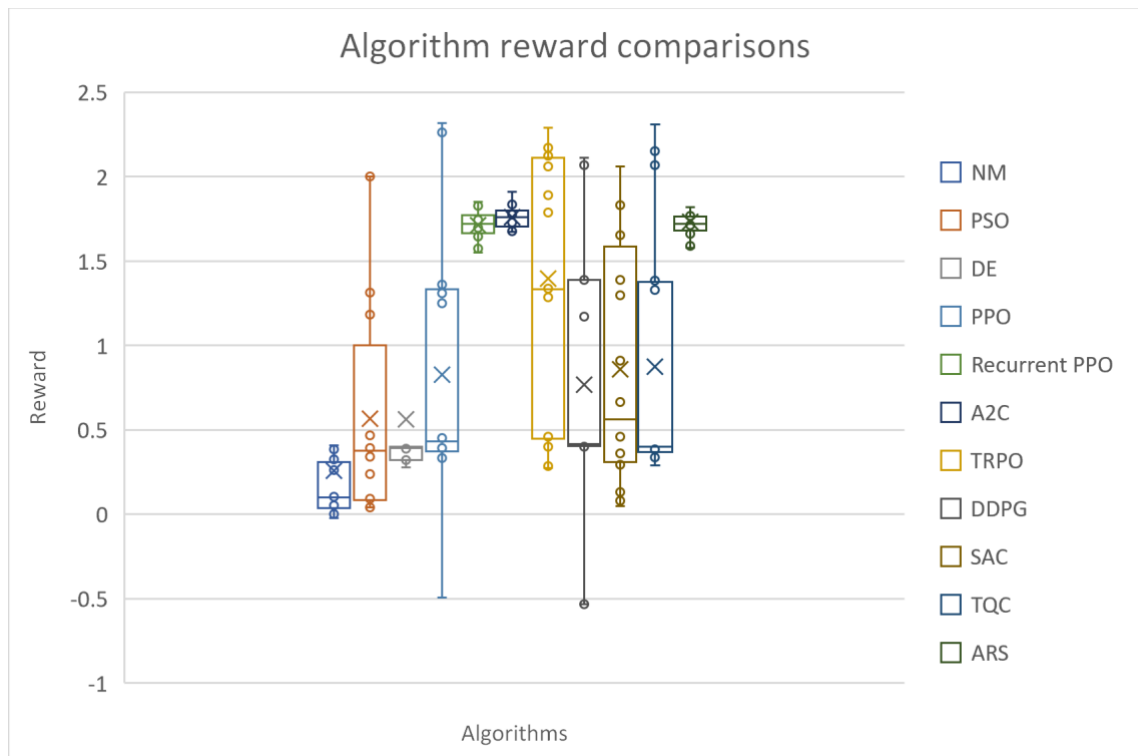


Figure 6-3 Algorithm performance - Jowitt & Xu

The processing time provides context on the computational effort needed to solve the burst scenarios through valve management. In figure 6-4, we display the different processing times for the benchmark optimisation algorithms, the DRL training times and the DRL test times. It is crucial to distinguish between training times required to develop the behavioural policy and test time required to solve the burst scenario. Throughout all test scenarios, it is clear that DRL algorithms decrease computational demand considerably especially when comparing DRL test times to the benchmark optimisation times. This further proves the ability of DRL algorithms to tackle complexity and long-term dependencies through feature extraction. Comparing the best performing algorithms from the previous figure yields a 197.6x increase in speed when deploying the A2C algorithm rather than the PSO algorithm. The fastest DRL test time was achieved by the A2C model only requiring 17.4 seconds followed by the DDPG model (17.8s) and the SAC model (18s). Consequently, all the DRL test times qualify them for real-time control as they signify the time taken to solve 72 timesteps over 3 episodes.

On the other hand, DRL training times show the time taken to run 20,000 timesteps developing the best behavioural policy. All DRL algorithms managed to reach their best reward within 20,000 training timesteps and increasing them showed minimal or no improvement. Training time depends mostly on the DRL methodology and the neural network architecture hence why more complex algorithms such as the distributional DRL algorithm (TQC) took the longest to

develop. Similar to the previous case studies, the second slowest training time was achieved by SAC. The fastest training time was awarded to ARS with 252s and the TRPO with 282s.

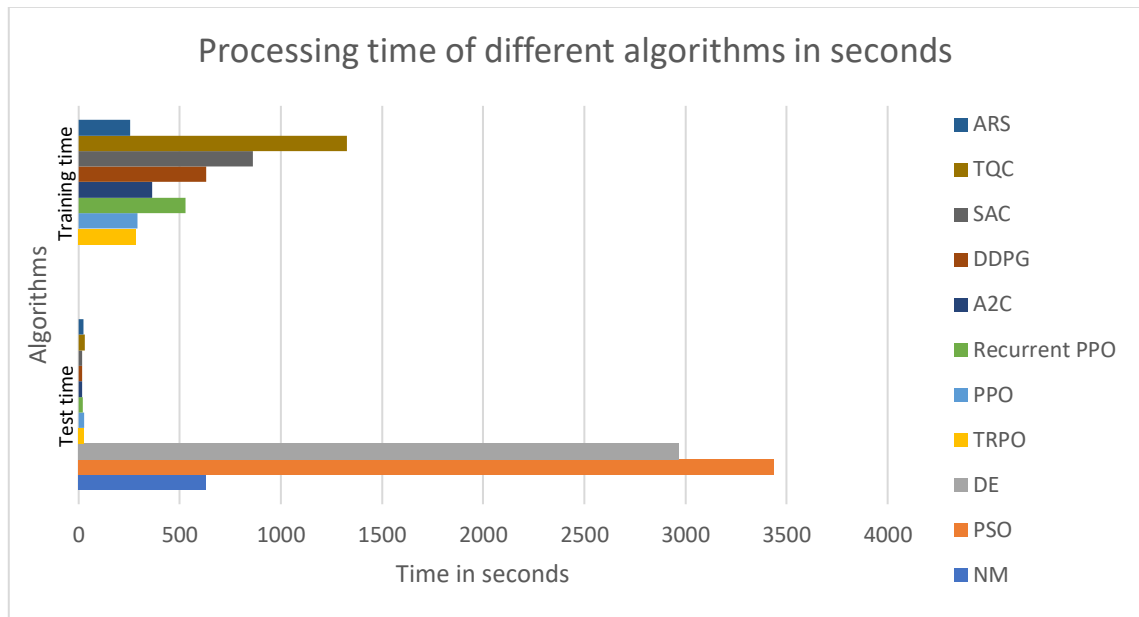


Figure 6-4 Algorithm speed - Jowitt & Xu network

Overall, the data collected from the case study can be summarised into the episodic results **table 6-2** below. Each algorithm's episodic performance in the three major sections of rewards accumulation, water loss and pressure violations are recorded alongside that carbon emissions reduced and total processing time. Furthermore, the best results are highlighted in bold to extenuate the most effective algorithm in each category. The A2C model yielded the highest average rewards at 42.15 and the fastest implementation (test) time at 17.4s making it the fastest and most effective algorithm for real time pressure management in this case study. A2C was also the highest performer in terms of water saved (58.46%) and carbon emissions reduction (5650kg). This was followed by Recurrent PPO (41) which improved greatly on the PPO rewards (20.43) and scored best in pressure violation minimisation with only 58 violations. Finally, the fastest algorithm in policy development was ARS due to its simpler methodology only requiring 254s.

Table 6-2 Key results - Jowitt & Xu

Algorithm	Average Reward	Average Water Saved (%)	Average Pressure Violations	Carbon Emissions Reduction (KgCO2)	Training time (s)	Test Time (s)
NM	6.791	12.68	62	1844	NA	631.8
PSO	17.23	29.16	112	159.8	NA	12909.4
DE	15.41	32.52	203	6137	NA	2967
ARS	41.79	47.62	69	5621	254	24.6
SAC	21.82	31.88	126	4631	860	18
TQC	24.26	43.16	241	7434	1326	30.4
TRPO	33.29	47.01	244	7252	282	27
PPO	20.43	40.10	339	7602	292	26.6
Recurrent PPO	41.00	42.28	58	5490	529	19.9
DDPG	18.54	38.57	293	7563	631	17.8
A2C	42.15	58.46	67	5650	363	17.4

6.2.2. Discussions

In this experiment, optimisation algorithms were developed and tested on the standard test network Jowitt & Xu. The network model consists of 22 nodes and 3 randomised bursts to be solved using three optimally placed PRVs. The optimisation algorithms were rewarded and judged based on their ability to minimise leakage rate and pressure violations due to leakage. Comparisons are made mostly based on the algorithms' overall performance and speed which are the two motivators to developing DRL algorithms for real-time control of pressure in WDNs.

Performance

The performance of the DRL algorithms consists of rewards calculated from minimised pressure violations and % of water saved. The associated carbon emissions reduction through leakage reductions are included to highlight the environmental effect of the optimisation. Overall, the DRL models greatly outperform the benchmarks in managing the effects of random burst events. The lowest performing algorithm was the Nelder Mead optimisation that only marginally improved that state of the network yielding an average episodic reward of

6.791 by saving 12.68% of leakage water. Following that, is a close battle between DE which was rewarded 15.41 and PSO at 17.23. Both algorithms gathered rewards through opposite methods. While DE focused on minimising leakage on the expense of pressure violation, PSO scored lower in leakage reduction but improved the number of pressure violations by resolving an extra 91 violations. The benchmarks' lower performance is directly proportional to the accuracy of the model they are trained on hence why randomising the burst nodes lead to inaccuracies. Therefore, it is difficult to operate traditional optimisation methods to the burst scenario as it requires an accurate predictive model due to its inability to tackle the inherent randomness of WDN management.

Otherwise, DRL models are trained on this random burst environment allowing them to expect randomness from the WDN test scenario. Using the observation space, which consists of a list of nodal pressures, DRL models adjusted the weights of their deep neural network to store the best policy for selecting PRV settings. This method allows the DRL models to establish a connection between the bursts' effects on the observation space and the corresponding valve to monitor/change in the action space. This process highlights the importance of building optimisation algorithms that can react to real data from experience to expect randomness without compromising performance. Hence why, the DRL algorithms tested in this experiment massively outperformed the benchmark alternatives.

In the policy-driven family, the ARS and Recurrent PPO models scored the highest with episodic rewards of 41.79 and 41.00 respectively. This was followed by TRPO scoring in 33.29 and PPO scoring 20.43 for episodic rewards. On average the policy driven DRL models performed better than the hybrid models (except for A2C) and distributional DRL. This can be explained by policy-driven methods' ability to navigate the exploration-exploitation trade-off through stochastically selecting actions making it the more attractive option where exploration was critical. In addition, the direct alteration of the behavioural policy allows policy driven models to be more flexible in capturing and adapting to changes in the environment. This allows the models to learn a self-adaptive policy that can deal with the randomness of the burst scenarios. A closer look on policy-driven algorithms shows ARS as the best performer in leakage reduction (47.62% water saved) and carbon emissions reduction (5621kg CO₂) while Recurrent PPO has the lowest number of pressure violations (58). Both impressive achievements helped land the ARS and Recurrent PPO in the top 3 positions for DRL models reward performance. ARS's high performance can be attributed to its ability to randomly search across the parameterised policy hence allowing it the flexibility to explore better policies. On the other hand, the Recurrent PPO model improves greatly on its predecessor the

PPO algorithm through the use of recurrent neural networks (RNNs) namely long short-term memory nodes. This allowed it to accumulate information over time hence tackling the challenges brought by partial observability and capturing sequential dependencies.

Hybrid models include A2C, DDPG and SAC. Plagued by its sensitivity to non-stationary environments, DDPG could not perform at the same level as the other DRL algorithms resulting in 18.54 average episodic reward which was marginally better than the benchmark PSO algorithm. DDPG's slow adaptation of the target networks and deterministic policy gradients making it unsuitable for the burst scenario. Surprisingly, SAC's incorporation of an entropy term to encourage exploration did not wage as well as expected in the burst scenario leading to an episodic reward of 21.82. The best explanation to this could be a misleading critic neural network as the value function fails to include the stochasticity necessary to judge the actor due to the changing environment. This would also explain how using an advantage function in the A2C algorithm greatly improves the results reaching the best overall performance with an episodic reward of 42.15. The advantage function provides a measure of how good an action is in comparison to the average action for a given state providing more efficient exploration and a clearer credit assignment. A2C achieves this high reward by striking the best compromise between leakage reduction (58.46%) and minimising pressure violations (67) therefore improving on both ARS and Recurrent PPO's performances.

Finally, the truncated quantile critics (TQC) model provides the only distributional DRL model in the cohort. TQC builds on the SAC foundations with a probability distribution over the value function for every state rather than a single value. This provides an additional dimension to the critic's evaluation of the agent's action which helps tackle the changing environment. This minor change results in a 11% improvement on SAC's performance making TQC's average episodic reward 24.26.

A notable observation is the superiority of on-policy DRL algorithms (A2C, PPO, Recurrent PPO and TRPO) over the off-policy algorithms (DDPG, SAC and TQC) in reward performance with the exception of ARS which performed well as an off-policy method. This is a testament to on-policy methods' suitability for online learning as they depend on experience to build their models while off-policy algorithms deploy replay memory methods that are often less stable. Therefore, on-policy methods are more suitable for dynamic environments as the case study requires due to the introduction of random burst nodes.

Speed

Evaluating processing times for the optimisation algorithms provides insights on real-time control possibilities thus increasing its importance. Real-time pressure control would significantly increase the WDN's resilience to anomalies and extreme conditions as it would be able to respond promptly. This will consequently improve overall pressure management and leakage control. Benchmark optimisation algorithms are compared to the DRL cohort for their test time which signifies the time required to complete 72 timesteps of the case study that model three sets of daily customer demand patterns. This comparison was made in **figure 6-4** and **table 6-2** where DRL clearly outperformed the benchmarks in solving the test case. The DRL algorithms' ability to use their developed policy to predict the best action simplifies the action selection process and minimises computational load. This is one of the biggest motivations to implement DRL for WDN pressure management and other real-world applications. As the slowest DRL algorithm (TQC) improves on the fastest benchmark algorithm's speed (NM) by a 20.8x mark up, it is safe to say that DRL algorithms are the best option for faster processing and better computational efficiency. TQC's demanding methodology requires it to collect a probability distribution of the value function hence the higher computational load and slower test time in comparison to other DRL algorithms. The fastest algorithm happens to be the best performing agent produced using A2C. This agent managed to implement its solution within 17.4s making it marginally faster than the DDPG agent at 17.8. In all cases, the DRL cohort is capable of real time control as it produces actions within the range of 0.242-0.422 seconds of receiving observations.

Separately, the DRL cohort is judged based on its training time. This signifies the time required to develop a policy within 20,000 timesteps of training data and is denoted as training time. Algorithms with simpler methodologies and neural network architectures tend to be easier to train such as ARS which conducts a direct random search on the parameterised policy equation. This yields the fastest training time of 254s. Training times can highlight which algorithms would be easiest to develop and re-train. This could be beneficial when considering a continuous improvement loop to ensure that seasonality doesn't affect the relevance of the DRL algorithms. As expected, the distributional algorithm TQC requires the longest training time with 1326s to complete the training.

6.3. Northumbrian Water Network

System Zone 08 (SZ08) hydraulic model serves the purpose of applying the optimisation algorithms to a realistic case study based on data retrieved from Northumbrian Water. This network surpasses the Jowitt & Xu benchmark in size, elevations, complexity, variety of

components and variety of customer demand patterns making it a more difficult network to optimise. In addition, the valve location and quantity are far less optimal for the size of the network, yet it models the real-life setup. A summary of network components can be found earlier in **table 5-5** and the network architecture is displayed in **figures 5-8** and **5-9**. As mentioned in the methodology section 6.1., the number of burst nodes will match the number of pressure valves in the network to provide a fair comparison of pressure management capability. Therefore, the SZ08 is initialised with 32 randomised bursts with the emitter coefficient/magnitude of three. To complete the leakage equation an exponent of 1.18 was used to align the research with the best practices in literature (Araujo *et al.*, 2006; Saldarriaga and Salcedo, 2015b). Experimenting the burst leakage case study with a real network such as SZ08 will also provide insights on the scalability of the models used and whether real-world application is a possibility. It is expected that, similar to the previous case study, the complexity of the network will push the DRL models slightly out of the zone of real-time control. Previous results may suggest that more stable DRL models perform better in this complex problem however, due to the unique nature of burst leakage, exploration and flexibility could be the key to better performances.

Using this hydraulic model, the environment will initialise the 32 random bursts for the network to solve. Optimisation algorithms will be developed or trained using the randomly amended environment but tested on the SZ08 model with burst nodes at the following nodes:

'N9968633'	'N12709278'	'N12717930'	'N10210106'	'N16111054'	'N10091680'
'N10094919'	'N12718004'	'N10208428'	'N12707262'	'N16119942'	'R16111847'
'NX34291839'	'N10205128'	'N10092063'	'N10205817'	'N10094345'	'N16109247'
'N10207834'	'N9963619'	'N10205153'	'N10208426'	'R10091961'	'N10028382'
'N13656814'	'N13656868'	'N9968663'	'N16111939'	'R16119652'	'RX33753118'
'N13663171'	'N10130639'				

These nodes were chosen using the same randomiser present in the environment. Due to the size of the network architecture and data, it is impossible to display all properties in a table therefore the input file will be included in the appendix. **Figure 6-5** below shows the distribution of these bursts (red) across the SZ08 network produced by the WDN-DRL environment.

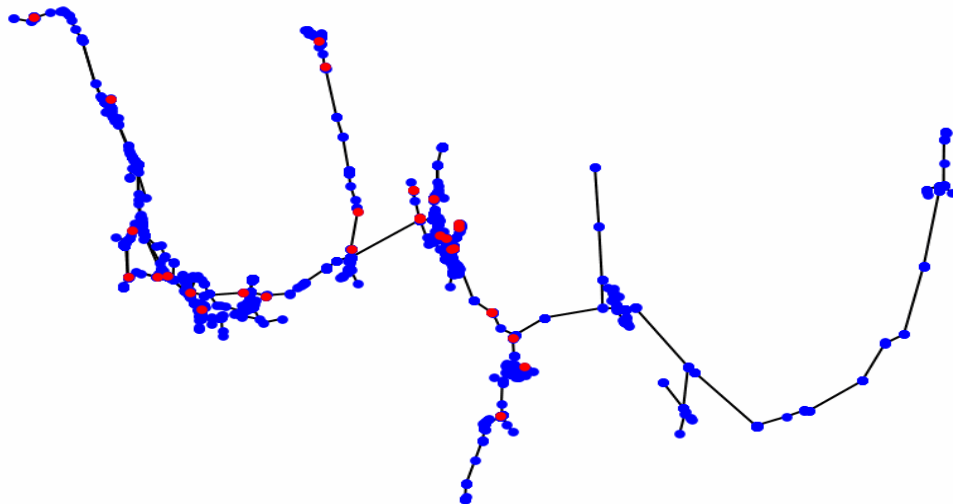


Figure 6-5 SZ08 network architecture with bursts in red

Finally, several DRL models were trained with the sole difference of reward ratios to highlight the most beneficial reward scales that should be assigned to the reward function. The models' performances are judged equally on their ability to balance the main objectives of leakage minimisation and pressure management. For the burst leakage case study, it was crucial to repeat this simulation until the best reward scale ratios is clear. As is clear in **figure 6-5**, most reward scales obtain similar episodic penalties apart from the ratio 1:4 favouring the pressure violation objective. The 1:4 scale ratio's penalty is not vastly different with only 12 points higher which is relatively miniscule. The results of the experiment showed that a ratio of 5:1 favouring leakage reduction strikes the best trade-off between the two objectives.

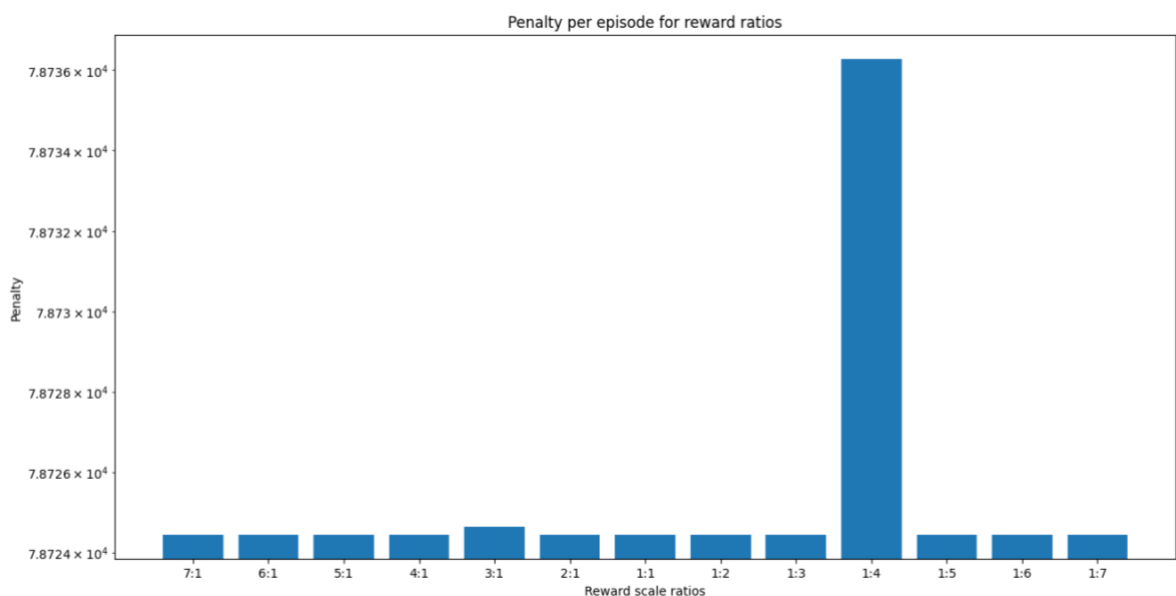


Figure 6-6 Episodic penalty for different reward scales.

After initial evaluations of the SZ08 model, it was clear that the reward function had to be slightly modified to better express the effects of leakage at low nodal pressures (0-1m). Low nodal pressures create highly negative rewards when leaks are increased in certain nodes (69 nodes, **figure 6-7a**). This exaggerated the extent of the penalty thus overshadowing the positive rewards of the optimised valve settings. The new reward function includes a hyperbolic tan function (tanh) to keep the leakage portion of the rewards between -1 to 1 shown in **figure 6-7b**. The new reward function is represented below in **equation 6-1**.

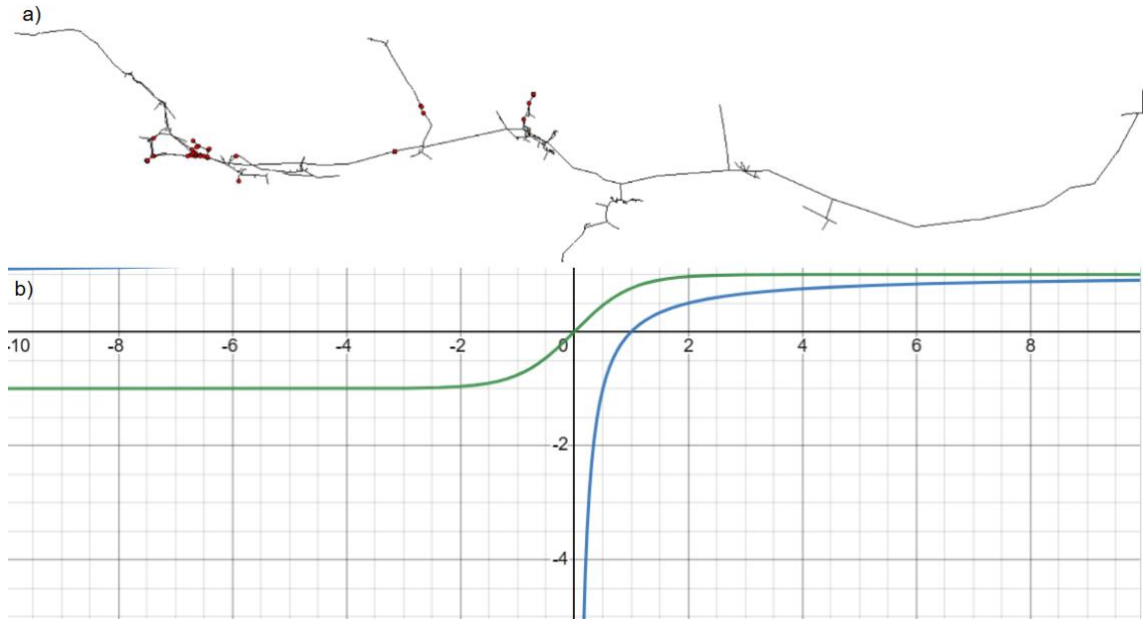


Figure 6-7a) SZ08 nodes with pressures lower than 1m. **b)** Old function (blue); new tanh() function (green).

$$R = \sum_{j=1}^M (v_{after} - v_{before}) \cdot scale\ 1 + \tanh\left(\frac{Q_{la} - Q_{lb}}{Q_{lb}}\right) \cdot scale\ 2 \quad (6-1)$$

6.3.1. Results

The SZ08 model included many complexities that made solving the case study inherently difficult. Introducing random bursts throughout the network provided was one example. The optimisation algorithms were evaluated using the reward function above (**eq. 6-1**) for three episodes lasting 24 steps each. The test scenario introduced bursts in the locations shown in **figure 6-5**. To highlight the algorithms' performances, the box plot is displayed in **figure 6-8** where the hourly rewards at each step are marked with an 'o' and the average reward is denoted by a 'x'.

Some optimisation algorithms were unable to optimise the case study causing negative rewards which hints to valve action lowering the network performance (through higher leakage rates or more pressure violations). These underperforming algorithms include the

benchmarked PSO (-8.53) and NM (-12.0) alongside the DRL algorithms DDPG (-14.7), SAC (-19.1) and ARS (-14.9). In contrast, differential evolution performed better with an average of 20.1. The DE boxplot hints that whilst the algorithm managed to optimise the test case, the interquartile range of the box shows that the DE algorithm has not fully grasped the best policy to control the pressure valves resulting in some negative rewards and some highly positive rewards. The best overall rewards were achieved using the A2C model (83.0) followed closely by the Recurrent PPO model (76.6) and TRPO (75.9). The distributional DRL algorithm, TQC, was the only high performing off-policy algorithm (72.5) followed by the on-policy PPO model (69.2). Like the Jowitt & Xu network, the DRL algorithms have outperformed the benchmark algorithms because they built a more resilient policy that has adapted to the randomness of the bursts. A clear trend shows that on-policy DRL algorithms (A2C, PPO, Recurrent PPO, TRPO) have developed more effective policies than their off-policy alternatives (DDPG, ARS, SAC).

A more detailed outlook on the step-by-step performance of the agents was displayed in **figure 6-9**. The line graph represents the rewards collected through the 3 episodes of the test scenario. The plots match the previous figure with on-policy algorithms massively outperforming the off-policy and benchmarked algorithms. The line graph also highlights DE's wide range of rewards shown in grey. Another noticeable trend is the higher rewards found during the minimum night flow hours between 3 to 5am. This is due to the higher average pressure during these times and hence the lower probability of invoking the low-pressure limit (less than 10m). Furthermore, minimising pressure during MNF hours is more possible resulting in greater leakage reduction.

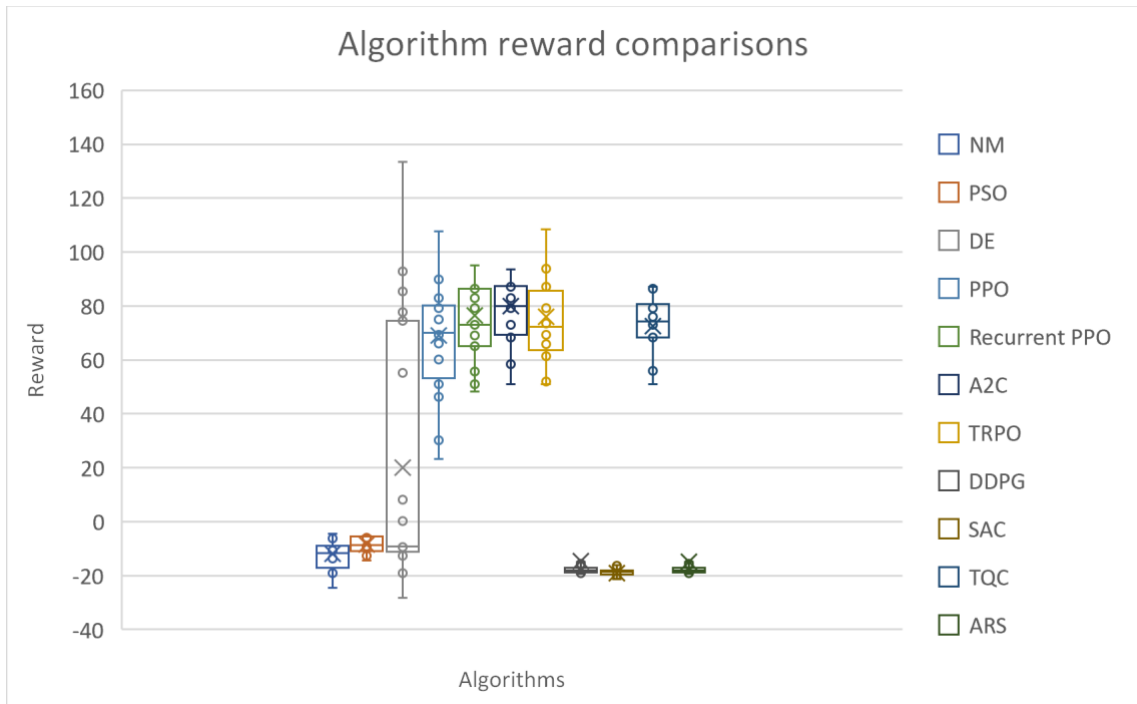


Figure 6-8 Algorithm performance - SZ08

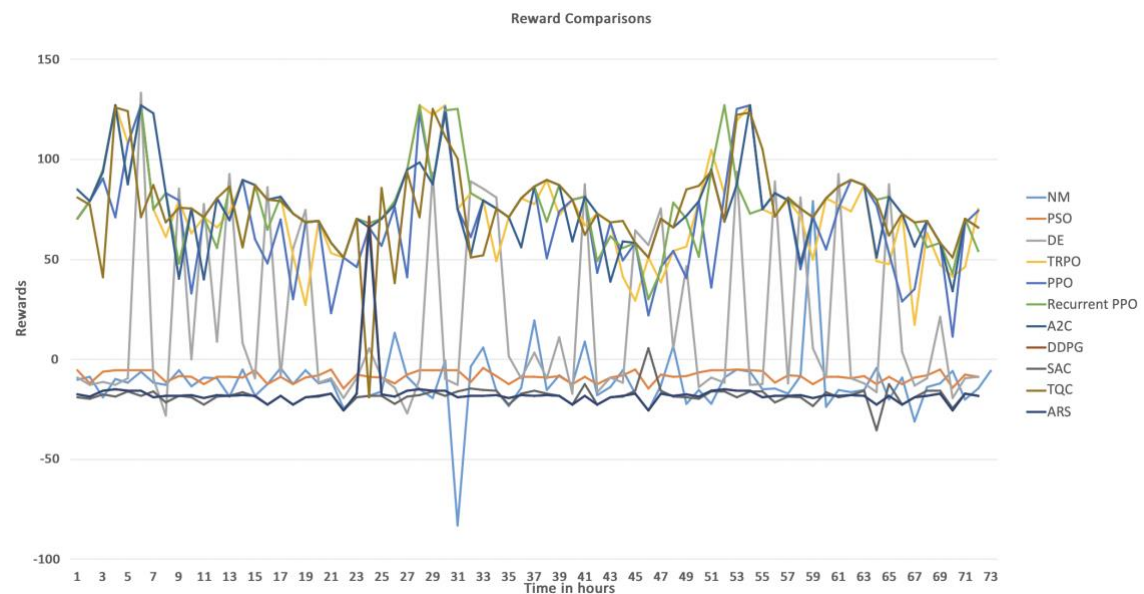


Figure 6-9 Reward time comparison – SZ08

Evaluating the processing speed of the different algorithms provides insight into their computational efficiency. The speed is measured in the time required for the algorithms to solve the 3 episodes of the test scenario (test time) and the time taken for the DRL algorithms to develop their policies using 20,000 timesteps (train time) as shown in **figure 6-10**. The training time can be used to assess which DRL models are easiest to build and re-train to avoid inaccuracies arising from seasonality and trends in water network operations. Models with smaller neural networks generally have faster training times as they are less computationally

demanding. In addition, processes that include distributional value functions such as TQC (6113s) are more computationally expensive requiring longer training times. SAC had the second longest training time due to its large neural network architecture at 5588s to develop its policy. Recurrent PPO needed 4519s to train its models which is slightly longer than the other DRL algorithms due to the LSTM modules adopted in the neural networks. The rest of the models have similar training times ranging from 3083s to 3672s.

In contrast, the benchmark algorithms act immediately without building a policy thus explaining the absence of any training time figures. These algorithms are compared on their implementation time otherwise known as their test time. As shown in **figure 6-9**, the DRL algorithms are much faster than their benchmark alternatives in providing valve settings and solving the test scenario as they rely on their predetermined policy to do so. On the other hand, NM (10983s), DE (10753s), and PSO (7627s) find the optimised action through multiple iterations at each timestep hence elongating the processing time. The DRL cohort demonstrate their fast-processing times with a speed mark-up ranging from 8.87-18.5x in comparison to the benchmarked algorithms. The fastest DRL algorithm is the PPO only requiring 594 seconds to complete 3 episodes totalling 72 timesteps with all the data renders mentioned in section 4.2.5. This corresponds to 8.25 seconds per timesteps which doesn't qualify the method as real-time. Removing the data visualisation tasks would surely cut down the processing times bringing it closer to real-time performance. The slowest implementation was the SAC which completed the test case in 860s.

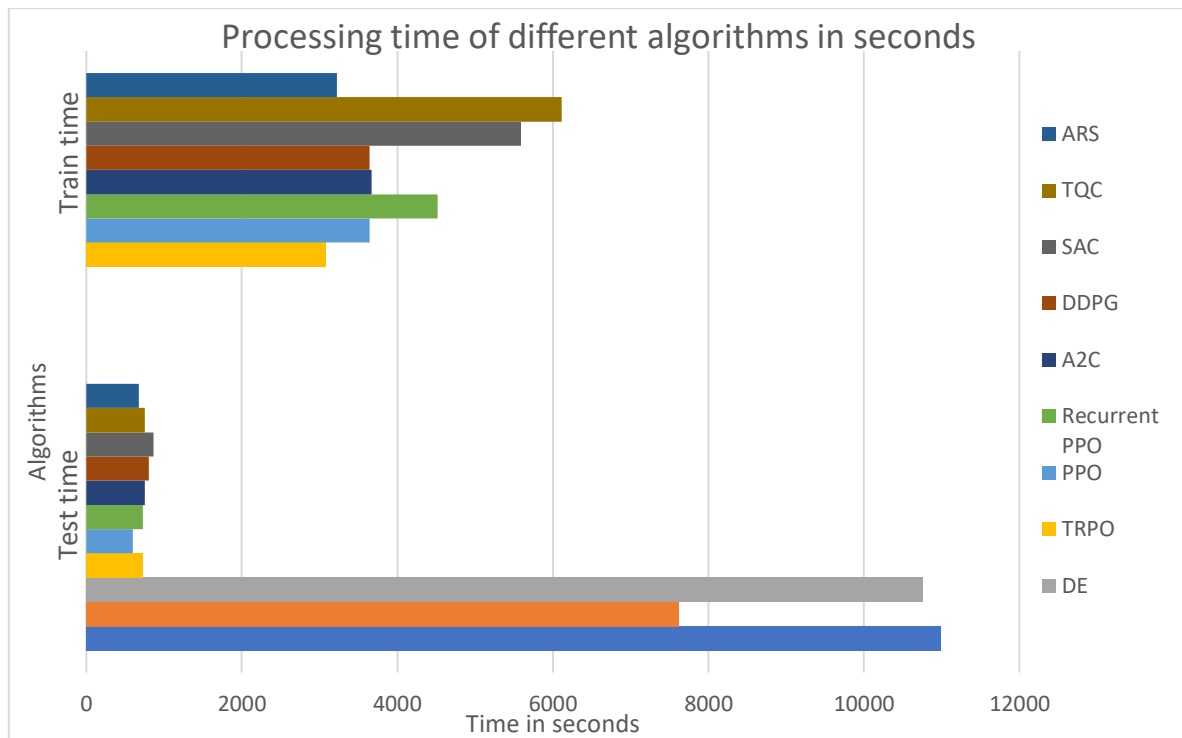


Figure 6-10 Algorithm speed - SZ08

To summarise the results, **table 6-3** was formed with the main metrics produced from the tests runs. This includes the average episodic rewards, average episodic water saved percentage, average episodic pressure violations, average episodic carbon emissions reduction and total processing time (train and test). The agents are assessed on their ability to maximise their episodic rewards by minimising pressure violations and maximising the % of water saved and pressure. Reducing leakage can be converted to useful carbon emission figures that help highlight the environmental impact of pressure management measures in Kg of CO₂. The best performer in every category was marked in bold. Despite SAC being the most effective in terms of leakage reduction (7.169%) and carbon emissions reduction (2492 KgCO₂), the agent achieves this by disregarding pressure violations resulting in the highest number of episodic pressure violations (17002) and a negative average episodic reward (-438.9). In contrast, the Recurrent PPO agent handles the trade-off between the reward objectives better by achieving the lowest number of pressure violations (14688) which results in the best episodic rewards at 1842. The algorithm with the best training time was TRPO only needing 3083 seconds to develop the policy while PPO had the fastest test time at 594 seconds.

Table 6-3 Key results - SZ08

Algorithm	Average Reward	Average Water Saved (%)	Average Pressure Violations	Carbon Emissions Reduction (KgCO ₂)	Training Time (s)	Test Time (s)
NM	-267.9	4.853	16817	1724	NA	10983
PSO	-207.0	4.967	16744	1778	NA	7627
DE	436.8	4.658	16105	1662	NA	10753
ARS	-413.3	7.077	16977	2426	3222	675
SAC	-438.9	7.169	17002	2492	5588	860
TQC	1828	5.878	14714	2027	6113	754
TRPO	1774	6.041	14771	2076	3083	724
PPO	1652	5.431	14898	1893	3644	594
Recurrent PPO	1842	5.793	14688	1999	4519	727
DDPG	-411.5	7.098	16978	2426	3644	807
A2C	1825	6.382	14712	2170	3672	756

6.3.2. Discussions

The test case scenario was developed to investigate the performance of 11 algorithms (3 benchmarks and 8 deep reinforcement learning). Running this experiment highlighted the algorithms' ability to minimise leakage and pressure violations within the real network model (SZ08). Comparisons are drawn based on the algorithms' performances and their processing speed in this section.

Performance

Using the results presented in **figures 6-8 & 6-9** and the key results in **table 6-3**, we discuss the performances of the algorithms tested. The reward function used to evaluate the models was altered to better suit the SZ08 network in **equation 6-1**. The performance of the algorithms varied greatly with some unable to optimise the test case due to its complexity. The results table showed how the best algorithms were the ones that focused on balancing the two objectives rather than minimising leakage which resulted in negative rewards. A closer look on **figure 6-9** shows the temporal rewards during the 72 timesteps of the tests. This line graph highlighted a clear pattern of higher rewards during lower customer demand times of the day. During the period of 3:00 to 5:00 of each day, night flow is at its lowest and nodal pressure rises to its highest. This allows agents to make more noticeable changes to the network by

reducing these nodal pressures and minimising leakage. In addition, the lower pressures do not invoke as many violations as other periods of the day since the average pressure is further from the low-pressure limits. The figure also shows the gap between the high performing algorithms and the underperforming algorithms. A better representation of this was in **figure 6-8** that displayed the rewards (and penalties) obtained by the tests in a box plot. **Figure 6-8** shows six algorithms that were able to optimise the test scenario (PPO, Recurrent PPO, A2C, TRPO, TQC and the benchmark DE). The other algorithms (DDPG, SAC, ARS, and the benchmarks NM and PSO) have failed to develop a useful policy and resulted in heavy penalties during the test scenarios.

The benchmark algorithms included particle swarm optimisation, nelder mead and differential evolution which mostly performed poorly in the test scenarios. The PSO and NM algorithms ranked 7th and 8th overall with episodic rewards of -207 and -269.7 respectively. Their negative rewards signify either an increase in leakage or pressure violations or both. In this case, an attempt to minimise leakage resulted in more pressure violations (16817 for NM and 16744 for PSO). The failure on the pressure management objective resulted in the negative performance which shows the algorithms' failure on navigating the trade-off between the two objectives. In comparison, the DE algorithm has managed to tackle this issue better resulting in a positive reward (436.8). The boxplot (**figure 6-8**) shows the randomness of the algorithm's reward collection insinuating the algorithm's convergence at a local minimum. DE achieves this superior reward by placing more emphasis on reducing pressure violations (1662) at the expense of the lowest leakage reduction with only 4.658% of the water saved. The general performance of the benchmark algorithms was poorer than the DRL cohort stemming from their inability to build intelligence and bespoke policies. The benchmark algorithms use the same methods to obtain their optimised results regardless of the problem setup therefore they are at a disadvantage when applied to dynamic environments such as the burst leakage case study. Practically, these algorithms rely on model predictive methods to solve the problem rather than real data making them reliant on the quality of the predictive data.

The DRL algorithms have had mixed performances with on-policy algorithms highly outperforming their off-policy alternatives. In the hybrid family, the on-policy A2C algorithm managed to develop the overall second-best performance with an episodic reward of 1825. A2C improves greatly on the DE benchmark by simultaneously reducing the leakage (6.382% water saved) and pressure violations (14712) showing a far superior policy. Evaluating **figure 6-9** shows room for improvements and tuning in A2C's performance despite it being one of the higher performers. Unlike A2C, the rest of the hybrid (DDPG and SAC) algorithms were off

policy. Off policy algorithms are generally less robust in dynamic environments as they rely on replay buffers as a learning strategy whilst on policy algorithms can learn online from real-time data. In order to collect rewards, DDPG and SAC emphasise leakage reduction at the expense of more pressure violations. As a result, both algorithms incurred heavy penalties. The DDPG agent failed to optimise the test case incurring a penalty of -411.5 which was lower than all three benchmarks (DE, PSO and NM). DDPG had the second highest water saved % with 7.098% and the second highest number of violations 16978. Despite SAC achieving the best leakage reduction with 7.169% water saved and the highest carbon emission reduction of 2492 Kg of CO₂; it has managed to get the worst overall performance with a penalty of -438.9. The penalty was solely due to the numerous pressure violations amounting to 17002 violations. In this scenario, it was clear that small policy changes could cause large changes in pressure violations making it an important focus in the reward trade-off.

The policy driven algorithms could also be split into on-policy (PPO, Recurrent PPO, TRPO) and off-policy (ARS). As expected, the on-policy algorithms far outperformed the off-policy alternative due to their ability to learn in dynamic environments. The on-policy algorithms also boast their stability and superior exploration strategy during training therefore reaching better trade-offs. The best overall episodic performance was achieved by the Recurrent PPO (1842) model that improved on its PPO (1652) predecessor through the inclusion of LSTM modules to the neural network. This minor change aided the neural network in building long term dependencies hence developing a better understanding of the case study. This allowed the Recurrent PPO model to develop resilience to the varying customer demand patterns forming a better overall policy. Recurrent PPO boasts the best performance with regards to the pressure management objective by minimising violations to 14688 in comparison to PPO which had 14898 violations. Both algorithms used lower leakage reduction to achieve this result with Recurrent PPO saving 5.793% and PPO saving 5.431% of leakage water. This was a significant drop from TRPO which managed to save 6.041% yet only scored a reward of 1774 due to its pressure violations of 14771. TRPO has been proven to handle the large scale of the SZ08 network in the previous chapter through the use of its trust region calculations however the randomness of the bursts required further knowledge of temporal dependencies which was exhibited by the Recurrent PPO algorithm. Despite the success of policy driven algorithms and ARS's general high performance in previous cases, the off-policy method resulted in a poor learning strategy. Unfortunately, that meant that ARS has incurred the overall second worst penalty of -413.3 by neglecting the pressure management objective. Whilst the algorithm

managed to achieve the best leakage and environmental performance (7.077%, 2426 Kg CO₂), it has incurred more penalties due to the numerous pressure violations (1677 violations).

Finally, a surprising result placed TQC as the only high performing off-policy algorithm. The TQC method relies on representing the value function to the critic as a distribution across states rather than a single value. This strategy allows the critic to gain a better understanding of how to evaluate the actions of the actor. Therefore, the distributional DRL hybrid algorithm manages to follow a better training trajectory leading to a higher performance (1828).

Speed

A crucial metric that was used to evaluate the optimisation algorithms was processing speed which was displayed in **figure 6-11** and **table 6-3** earlier. Due to the focus of real-time implementation and the dynamic behaviour of real network, speed was a major consideration. The algorithm speeds were assessed based on their implementation time (test time) and policy development time (training time). Evaluating the DRL algorithms for their training time provides an opportunity to highlight the models easier to build and amend. **Figure 6-11** displays bar graph of the training times which denote the time required to build a DRL model through 20,000 timesteps. The training times are mostly dictated by the deep neural network size and the DRL method needed to train the policy through gradient descent. TQC has the largest neural network size and a complicated DRL method to deploy a value distribution for the critic agent. As a result, it has landed the longest training time of 6113s. SAC has an identical neural network to TQC but does not require the value distribution to train the critic agent making it a simpler method and resulting in faster training (5588s). PPO, A2C and DDPG have similar processing times due to their smaller neural networks with training times of 3644s, 3672s and 3644s respectively. Including LSTM modules to PPO's neural network creates the Recurrent PPO therefore requiring an additional training time of 4519 seconds. ARS adopts a simple method to develop its policy through a random search over the policy parameters further reducing the computational demand and lowering its training time to 3222s. Finally, the fastest DRL algorithm to train its model was the TRPO algorithm that only required 3083 seconds to complete the 20,000 timesteps. TRPO's trust region equations help it converge faster when dealing with complex environments further allowing it to develop its policy more efficiently.

More importantly, algorithms are assessed on their ability to optimise the test scenario based on their test time. Test times are measured as the time required to solve 72 timesteps of the test scenario signifying three days' worth of customer demands. The main comparison lies

between the implementation time for DRL and non-DRL benchmarks as shown in figure 5-11. The benchmark algorithms were outperformed by the DRL cohort by providing an 8.87-18.5x speed mark-up. This is possible through DRL's ability to develop an overarching policy applicable to different test scenarios. PSO was the fastest benchmark algorithm to converge to a solution needing 7627s. This was followed by DE and NM which were the slowest two algorithms overall demanding 10753s and 10983s respectively. These benchmark methods are often paired with model predictive control (MPC) to estimate the best valve settings for the upcoming customer demands. Forecasting future models can lead to bias due to model inaccuracies whilst the ability to apply optimisation algorithms directly to the water networks bypasses these issues.

DRL models built through the training process can be applied to the test scenarios by using the current observations to suggest the next action through the build policy. Using this method, DRL algorithms can be applied directly to water network environments with superior test times. The fastest overall implementation was achieved by PPO by completing the test in 594 seconds while the slowest was SAC that took 860 seconds. Otherwise, the DRL cohort has achieved similar times between 724s and 754s.

6.4. Concluding Remarks

Burst leaks, detectable through modern techniques, are less frequent but pose significant challenges. The chapter introduces a methodology for testing the impact of pressure management on mitigating leakage through burst events. The case study evaluates different Deep Reinforcement Learning (DRL) models against benchmark algorithms, aiming to minimize water loss and pressure violations in a complex water distribution network.

During this case study, eight DRL algorithms and three benchmarks were deployed to optimise the pressure management of water distribution. The WDN used included a standardised test network (Jowitt & Xu) and a real world WDN model (SZ08). To highlight the algorithms' ability to reduce burst leakage and pressure violations two reward functions were used (**E.q. 3-3** and **e.q. 6-1**).

The burst leakage case study provided an actionable method to mitigate random burst events in water distribution networks. The study involved two experiments that focused on testing the scalability of the tested algorithms. A cohort of eight DRL algorithms and three benchmark algorithms are evaluated based on their ability to simultaneously reduce leakage and pressure violations. This particular case study is built with an extra layer of complexity due to the randomness of the burst locations making the environment more dynamic. Creating

relationships between the observation space and the action space becomes a more sensitive and crucial task.

Reviewing the results unveils a clear preference to the use of DRL algorithms to solve random burst events. DRL algorithms almost exclusively outperformed the benchmarks in both networks showing their ability to handle dynamic environments. Benchmarks were unable to optimise the case studies to the same degree due to their inability to develop an overarching policy. Furthermore, DRL models improve greatly in processing times making them the more beneficial option for real-time control. The benchmark algorithms were deemed unsuitable for real-time control due to their long implementation time explaining why they're mostly deployed in model predictive control methods. Real time control was achievable for smaller DMA-scale network however further work is required to improve processing speed for larger scale applications. Performance varied greatly within the DRL cohort as on-policy methods outperformed their off-policy counterparts. On-policy algorithms gain the advantage due to their stability with online learning and their superiority navigating dynamic environments. The only exception to that was ARS's performance in the Jowitt network and TQC's performance in the SZ08 network.

A comparison of the two networks displays the effects of scalability and complexity on algorithms performance. On the smaller scale, the DRL algorithms were better equipped to minimise leakage and manage pressure efficiently This is due to the optimal valve locations and the relative frequency of the pressure valves in the smaller benchmark. The sub-optimal locations left uncovered areas in the SZ08 network that are unaffected by valve action. Therefore, some areas within the SZ08 network cannot mitigate the burst events regardless of the optimised valve settings. In addition, the optimisation algorithms were better equipped to minimise leakage due to their coverage and control of the network. Stemming from that, it can be inferred that managing water networks on the DMA level is more likely to produce better results. Nevertheless, testing the DRL methods in the SZ08 network proved their scalability making them serious candidates for real life implementation.

7. Conclusions

In the process of advancing leakage management in water distribution network, this research has narrowed down to a focus on prevention through pressure management. Through proper valve control, WDN can simultaneously minimise leakage in the network without violating OFWAT regulated pressure limits. A novel methodology that exploits the capabilities of DRL algorithms for WDN applications has been developed to tackle background and burst leakage. A cohort of eight DRL algorithms and three benchmark optimisation algorithms are implemented to case studies consisting of four experiments in total applied to two different networks. In this chapter, the overall conclusions of the research are drawn from the case studies and their results. Any assumptions made during the research are elicited followed by the limitations. Finally, recommendations for future research are proposed to aid with the development of the research further.

Chapter 2

Prior to developing the DRL algorithms or the environment, a literature set was produced to explain the necessity of leakage management in WDNs, and current practices tackle this problem. The main conclusions underscore the importance of diverse leakage assessment strategies including the Top-down, MNF analysis, and BABE methods, while advocating for a combination of methodologies for a more comprehensive assessment. Leakage detection involves inspection robotic platforms and non-intrusive hardware methods. However, advances in software methods such as data-driven leakage detection, especially in hybrid approaches with neural networks, show promise. Effective leakage control encompasses pressure management and asset management considering various impact factors. Considering pressure management requires emphasis on optimal valve placement and control. In this research, the path of leakage reduction through pressure management is explored further through the novel introduction of DRL algorithms for valve control.

Chapter 3

Hereby, a review on the application of DRL in all aspects of urban water systems was conducted to validate the novelty. In this section a comprehensive review of DRL methods coupled with a classification tree leads the path to DRL in UWS. This is followed by a review of current research and trends of DRL in different aspects of UWS. The main conclusions of this literature set highlight the impact of the use of deep neural networks for function approximation leading to improved scalability and resulted in many successes across simulated and real applications. Current DRL trends tackle high dimensional complexity by mimicking

human psychology and natural hierarchy structures. A novel classification tree was established to help new researchers navigate better. The application of DRL in the UWS is still developing yet it shows great promise to improve our current practices with water. Early efforts to benchmark DRL test beds and environments will aid the growth of this topic. Challenges of applying DRL to water systems include testing and validation through real life models.

Chapter 4

The aim to deploy deep reinforcement learning methods for the optimisation of pressure management in WDNs could not be realised without the RL environment. As the literature suggests, building the DRL environment was an integral part of the research as it dictates the interactions between the agent and the WDN. The environment used managed to depict the problem effectively and lead the agents towards optimisation. All the necessary sections required to establish the environment were covered in chapter 4. The conclusions drawn from this chapter were crucial to the development of the research and highlight important remarks for the implementation of DRL in WDNs. Creating the environment has allowed for the use of both DRL and non-DRL methods to optimise the WDN models while leveraging the hydraulic simulation capabilities of EPANET. Implementing environment design for real networks will require the hydraulic solver capabilities of EPANET or a similar hydraulic solver. Input files developed from utility data platforms can be used in the environment as shown in the SZ08 network. Furthermore, defining the action and observation spaces are major decisions that heavily influence the interaction between the agent and the problem. They should model the real-life engineering problem. Another major consideration is reward formulation. Selecting the reward function is a very sensitive task and should be designed iteratively to achieve the desired objectives. Training the DRL model and testing the agents would seem like a black box process if the environment were not equipped with the necessary data visualisation and logging tools. Hence, the necessity of rendering and reporting functions to assist with data-driven decision making. Finally, it is important to deploy a variety of DRL algorithms to test the suitability of each algorithm and investigate their performances.

Chapter 5

Experiments carried out to minimise background leakage in WDNs included the use of a small benchmark and a large real WDN model. As a result, background leakage was minimised in both models through PRV action led by DRL and non-DRL optimisation algorithms. The DRL algorithms proved their effectiveness in pressure management by achieving beneficial results that led to the following conclusions. The best DRL methods achieved 73.4% leakage

minimisation and 302.5 Kg of CO₂ reduction in the Jowitt & Xu network. In comparison, the best DRL methods performed less favourably on the SZ08 network where it reached a maximum of 0.693% leakage minimisation and 169.3 Kg CO₂ reduction in SZ08. It is clear that the model's performance on background leakage is highly reliant on valve locations and coverage.

Furthermore, significant improvement in processing speed was observed when utilising the DRL models in comparison to the benchmarks. The case studies also proved that DRL algorithms are capable of real time pressure management in the Jowitt & Xu network and near real-time pressure management in SZ08. The DRL models were able to navigate the trade-off between pressure management and leakage minimisation effectively. Despite DRL models performing less favourably than their benchmark alternatives, their ability to reduce computational load and work real-time makes them a favourable option for WDN operation. Real-time control in water distribution networks promises leakage and carbon reductions without violating pressure limits by allowing instant valve reactions to network changes. Experimenting with DRL hyperparameter tuning significantly improves algorithm performance, indicating room for improvement through hyperparameter optimization. Comparatively, hybrid methods were more effective for the smaller network (Jowitt & Xu), while policy-driven methods provided stability for the larger network (SZ08). Overall, DRL algorithms show promise for real-time control at a District Metered Area (DMA) level with low computational load and high rewards. On the other hand, valve locations in the real network were sub optimal making it more difficult to manage and compromised the leakage reduction capabilities.

Chapter 6

The burst leakage case study evaluated eight Deep Reinforcement Learning (DRL) algorithms and three benchmark algorithms for mitigating random burst events in water distribution networks. By controlling valve settings in the small-scale test network (Jowitt & Xu) and the large-scale real network (SZ08), the DRL cohort was compared to benchmark optimisation algorithms on their ability to minimise leakage and pressure violations. The diversity of network scales and complexity answers questions on the stability and scalability of the models while presenting a solution for a real-life scenario. Randomised burst locations make the environment dynamic and more difficult to solve hence emphasising the importance of data-driven decision-making. The burst scenario unveiled several key conclusions. Almost all DRL algorithms, particularly the on-policy methods, outperformed the benchmarks in navigating

the dynamic environment resulting in lower leakage and pressure violations. The best DRL performances resulted in 47.60% reduction in leakage and 5650 Kg CO₂ emissions reduced in Jowitt & Xu network. Also produced a 5.793% decrease in leakage and 1999Kg CO₂ reduction in carbon emissions in the SZ08 network. Optimising the real network (SZ08) proved more challenging due to the sub optimal valve placement making it more difficult to manage certain areas in the network. Nevertheless, DRL models exhibited improved test times through their function approximation capabilities making them more suitable for real-time control. On-policy DRL algorithms were more capable of learning from online data making them perform better than their off-policy counterparts. The only high performing off-policy algorithm (TQC) was materialised through distributional DRL proving how using a value distribution can improve results. By comparing the PPO and Recurrent PPO performances, it is apparent that introducing LSTM modules to the neural network improves the ability to navigate the dynamic environment. The scalability of the DRL methods was proved through their implementation in SZ08, however their performance decreased slightly in comparison to the smaller scale network.

In summary, several experiments have been conducted to test and validate the application of DRL algorithms in leakage reduction through pressure management. As a result, we have managed to develop insightful data visualisation figures and compare the performances of several optimisation algorithms (DRL and non-DRL) using a novel pythonic environment. The DRL methods have been proven to operate in real-time or near real-time by adapting its policy to meet the current conditions of the network therefore completing the project aims set in section 1.3. In addition, the DRL's performances in both case studies highlight their ability to handle uncertainties in dynamic environments in a scalable manner. This affirms our novelty in the simulated experiments. The accuracy of these results is therefore contingent on the accuracy of the hydraulic data used to create the case studies and will need to be re-assessed before real-life implementation. More limitations on the results are outlined in the following section.

7.1. Limitations

In order to provide a full view of the research, it is crucial to address the limitations of the work and how it could affect the results. In this section, we highlight some of the main limitations associated with the use of DRL in pressure valve control in WDNs. One of the main limitations unveiled in the pressure management the real SZ08 network was the sub optimal locations of the pressure valves. To achieve the best results, water networks must be fully controlled by their pressure valves however real networks are far from their optimal states. As

displayed in SZ08, there are often areas of the network that are unaffected by valve action signifying leakage vulnerable zones. This dependency on valve locations and coverage is natural but should be mitigated by ensuring enough valves are present in optimal locations to maximise pressure control. Another method to increase pressure management capabilities would be to include the pumps as part of the control system. Coordinating between the pumps and valves can help improve pressure control in the network.

Another limitation of this study is its reliance on clear data. Data-driven optimisation is very insightful nevertheless it requires sensor data across the entire network. Water networks vary in their data availability and data quality which could limit the usability of DRL algorithms in WDNs. Therefore, this study is best applied to WDNs that have established a coherent data pipeline and are looking to expand their pressure management facilities. Consequently, it is important to build accurate hydraulic models that can be used to build the DRL agents. Well-developed DRL models also tend to be quite sensitive to erroneous observation data which could falsely trigger harmful actions by the pressure valves. The DRL input data must be cleaned and tested for accuracy to ensure that it represents the current state of the network.

Furthermore, the application of DRL requires reliability evaluations before being deployed on WDNs. In any engineering application, it is necessary to ensure that the optimisation algorithm won't endanger the customers. In this instance, agents need to ensure that water supply remains uninterrupted without affecting asset life or risking future bursts. These concerns were addressed by (Tian, Liao, Zhi, *et al.*, 2022) where the authors devised a 'voting' method to improve reliability. Water distribution networks are subject to daily and seasonal changes that will undoubtedly influence the performance of the DRL models. While the DRL algorithms were proven to deal with randomness in the observation data, seasonal changes might require re-training of the models and further policy development. This could be achieved through a continuous integration/deployment (CI/CD) pipeline for the DRL models which automates the deployment of newer, more suitable models.

Limitations also include the effect of the DRL algorithm on other objectives of the WDN. While this study is focused on using DRL algorithms for the purposes of pressure management and leakage reduction, other objectives could be affected by the pressure control hence why it is necessary to include these considerations in DRL algorithm design. To avoid this, any relevant objectives should be included in the reward formulation design to ensure that the agents are trained with a complete picture of the desired behaviour. Complex model design is not limited to the selection of the reward function but includes DRL sensitivity to hyperparameters and

neural network architecture. The design of DRL algorithms involve many decisions including various options for neural network architectures, optimisers, activation functions, pre-training techniques, and hyperparameters. The complexity of making these design choices require careful consideration and experimentation. Generalisation of the DRL models is limited as the policy developed for one network may not necessarily work for another therefore it is important to develop a separate model for each network. On another hand, the option for transfer learning between the neural networks is valid as that could help train models from different networks.

7.2. Assumptions

To carry out the experiments detailed in this study, several assumptions had been made. This is to abstract the leakage problem enough to make it solvable yet not affect the validity of the work. In reality, the leakage problem involves several dependencies that are out of the scope of this research so, to create a pressure management focused study, the following assumptions were made.

- The pressure-leakage relationship used in EPANET and denoted in **equation 2-9**. This equation describes leakage as a function of pressure and has been widely accepted in the research literature (Lambert, 2001; Thornton, 2003; Thornton and Lambert, 2005). In this equation the leakage exponent, n , was assumed to be 1.18 to align this study with the wider research community as this is the accepted value as shown in (Araujo *et al.*, 2006; Saldarriaga and Salcedo, 2015b).
- Another assumption made based on literature was the background leakage limit. Using UK based figures from (García and Cabrera, 2007), we assume that the leakage coefficient between 0 to 0.196 signify background leakage and coefficients above 0.196 signify burst events.
- On the other hand, the leakage coefficients, also known as nodal emitter coefficients, represent the state and length of the neighbouring pipes. In order to estimate background leakage in the SZ08 model we assume that the state of the pipes is identical and the deciding factor for leakage is the length of the pipe. Hence why, **equation 5-1** was used to estimate background leakage across the nodes. For the Jowitt & Xu network, these emitter coefficients were included in the benchmark.

This study included more general assumptions related to the methodology. These assumptions include.

- Hydraulic models used are considered the digital twin of the real networks. When implemented on the water network, the DRL algorithms will be acting based on real data making it less susceptible to discrepancies in the model. For the purposes of this study, a real-life implementation was not possible as there are no test networks available in the UK. The first test network will be built by Northumbrian Water and expected to be ready by 2025.
- Carbon emissions conversion factors were used from the government publication for 2023 (Department for Energy Security and Net Zero, 2023). The assumption made is that all the relevant carbon emissions are a result of leakage where in practice minimising leakage effects pumping and other network properties that might decrease carbon emissions further. For the scope of this study, only the direct carbon emissions are considered (scope 1).
- When selecting the action space, it was necessary to assume that it is continuous to allow the agents to roam the search space freely and equate the DRL agents to their benchmark alternatives. The assumption made was that pressure valves are able to process precise settings provided by the agent whereas in practice some control errors can be expected.

7.3. Recommendations for Future work

Beyond this research's scope, there is room for development. In this thesis, we take the first steps in deploying deep reinforcement learning for leakage reduction in water distribution networks yet the path to a fully realised system requires further work. To conclude this research, we recommend future research to include the following.

As this field continues to grow, so does the need to benchmark case studies and environments. Therefore, it would be beneficial to collate and benchmark the DRL environments created to solve certain problems withing UWSs. For example, the environment created for this research can prove as a useful benchmark to train DRL agents for leakage prevention or (Hajgató, Paál and Gyires-Tóth, 2020)'s environment for pump control. It is important to intensify DRL research in leakage management applications. The management of in pipe inspection robots can be controlled through DRL algorithms or even a prediction agent can be trained to classify leaks and leak locations. These are a few examples regarding the possibilities of DRL in leakage management. From an engineering perspective, it is crucial that researchers exploring this topic confirm the validity of the application through live data or ground truth models. To achieve that, DRL algorithms should be built with reliability and scalability in mind.

Different training methods such as Hindsight Experience Replay (HER), imitation learning and inverse RL should be investigated for their effectiveness in improving agent performances. These are among many tools that researchers should experiment with to further improve the results of DRL algorithms in mitigating leakage. Furthermore, researchers should aim to optimise neural networks through trialling different optimisers, activation functions, LSTM modules, architectures and more. It is proven the graph neural networks (GNNs) are effective in emulating the hydraulic behaviour of WDNs (Fan, Zhang and Yu, 2022) therefore it is safe to assume that they would be effective in creating DRL agents. There are many methods prevalent in the industrial application of DRL agents which could easily be repurposed for water distribution hence why researchers are encouraged to delve into DRL literature. It is adamant from the case studies that multi agent DRL could provide the resilience required to monitor DMAs separately and interactively ensuring that each network is optimised with an open loop between the DMAs. This method often incurs higher computational loads due to the complexities associated with multiple agents and the necessary environment modifications.

Incorporating other pressure influencing network elements such as pumps and air valves would add to the capabilities of the pressure control. The simplest way to achieve this is to add the new elements to the action space controlled by a single agent. This will require slight modifications to the environment to ensure that the different actuators can be managed simultaneously and account for the different nature of pumps to the pressure valves. Furthermore, variable speed pumps and fixed speed pumps constitute two different behaviours which should be considered when creating this advanced pressure system. With fixed speed pump actions could represent zone scheduling through a binary on or off state while variable speed pump actions could represent different speed settings. Adding pumps to the pressure management system will improve the agent performance hydraulically and provide insight into energy efficiency achieved through pump operation. Customising the reward function to include energy consumption or carbon emissions would add extra objectives of energy efficiencies and carbon reductions to the environment.

The case studies used in this research displayed different sizes and topologies to highlight the scalability of this method. Nevertheless, more experimentation with WDNs from across the globe can provide more proof to the useability of DRL for real-time control. It was evident by the case studies that the pressure management capabilities are also limited by the network elements therefore further studies could focus on optimising the number of valves, their locations as well as their settings. This will help realise the true capabilities of pressure management in leakage prevention.

Further investigation on the benefits and challenges of the practical application of DRL control should be undertaken. For example, the water quality level can be compromised through contamination at leakage sites however through pressure control this can be reduced considerably. Flow reversals and its effects can also be investigated and accounted for in the reward function. In addition, it is recommended that a cost-benefit analysis should be conducted over an extended period, preferably a year. This is to ensure that the analysis includes seasonality changes and trends observed in the annual year. Doing so will aid in providing an accurate projection to the 5-year and 10-year costs and savings. Ideally, this analysis would be conducted by a water utility company as it requires access to sensitive data unavailable to the research community such as in-depth knowledge of the costings for water treatment, man-hours, turnover, and the size and frequency of penalties.

These recommendations can serve as a guide to lead the next steps of implementing DRL to leakage management in WDNs.

References

A.W.W.A. (2020) *Free Water Audit Software*, American Water Works Association. Available at: <https://www.awwa.org/Resources-Tools/Resource-Topics/Water-Loss-Control>. (Accessed: 30 May 2020).

Abduljabbar, R.; Dia, H.; Liyanage, S. and Bagloee, S. (2019) 'Applications of Artificial Intelligence in Transport: An Overview', *Sustainability 2019*, Vol. 11, Page 189, 11(1), p. 189. doi: 10.3390/SU11010189.

Abdulla, M. B. and Herzallah, R. (2015) 'Probabilistic multiple model neural network based leak detection system: Experimental study', *Journal of Loss Prevention in the Process Industries*, 36, pp. 30–38. doi: 10.1016/J.JLP.2015.05.009.

Abdulshaheed, A., Mustapha, F. and Ghavamian, A. (2017) 'A pressure-based method for monitoring leaks in a pipe distribution system: A Review', *Renewable and Sustainable Energy Reviews*, 69, pp. 902–911. doi: 10.1016/J.RSER.2016.08.024.

Achiam, J. (2020) 'Spinning Up Documentation Release'.

Adams, S., Cody, · Tyler and Beling, P. A. (2022) 'A survey of inverse reinforcement learning', *Artificial Intelligence Review*, 55(6), pp. 4307–4346. doi: 10.1007/s10462-021-10108-x.

Adedeji, Kazeem B; Hamam, Yskandar; Abe, Bolanle; Abu-Mahfouz, A. M. (2017) 'Leakage Detection Algorithm Integrating Water Distribution Networks Hydraulic Model', in *SimHydro 2017: Choosing the right model in applied hydraulics*. Sophia Antipolis, pp. 1–9.

Adedeji, Kazeem B; Hamam, Yskandar; Abe, Bolanle; Abu-Mahfouz, A. M. (2017) 'Towards Achieving a Reliable Leakage Detection and Localization Algorithm for Application in Water Piping Networks: An Overview', *IEEE Access*, 5, pp. 20272–20285. doi: 10.1109/access.2017.2752802.

Adedeji, Kazeem B; Hamam, Yskandar; Abe, Bolanle; Abu-Mahfouz, A. M.. (2018) 'Pressure Management Strategies for Water Loss Reduction in Large-Scale Water Piping Networks: A Review', in *Springer Water*. doi: 10.1007/978-981-10-7218-5_33.

Adnan, N. F.; Ghazal, M. F.; Amin, M. M; Hamat, A. M. A. (2015) 'Leak detection in gas pipeline by acoustic and signal processing - A review', *IOP Conference Series: Materials Science and Engineering*, 100(1), p. 012013. doi: 10.1088/1757-899X/100/1/012013.

Ahadi, M. and Bakhtiar, M. S. (2010) 'Leak detection in water-filled plastic pipes through the

application of tuned wavelet transforms to Acoustic Emission signals', *Applied Acoustics*, 71(7), pp. 634–639. doi: 10.1016/J.APACoust.2010.02.006.

Ahiablame, L. and Shakya, R. (2016) 'Modeling flood reduction effects of low impact development at a watershed scale', *Journal of Environmental Management*, 171, pp. 81–91. doi: 10.1016/J.JENVMAN.2016.01.036.

Akiba, T. *et al.* (2019) 'Optuna: A Next-generation Hyperparameter Optimization Framework', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631. doi: 10.1145/3292500.3330701.

Aksela, K., Aksela, M. and Vahala, R. (2009) 'Leakage detection in a real distribution network using a SOM', *Urban Water Journal*, 6(4), pp. 279–289. doi: 10.1080/15730620802673079.

Al-Adeeb, A. M. and Matti, M. A. (1984) 'Leaching corrosion of asbestos cement pipes', *International Journal of Cement Composites and Lightweight Concrete*, 6(4), pp. 233–240. doi: 10.1016/0262-5075(84)90018-6.

Al-Washali, T. *et al.* (2020) 'Assessment of water losses in distribution networks Methods, applications, uncertainties, and implications in intermittent supply', *Resources, Conservation & Recycling*, 152, p. 104515.

Al-Washali, T., Sharma, S. K. and Kennedy, M. D. (2018) 'Alternative Method for Nonrevenue Water Component Assessment', *Water Resources Planning and Management*, 144(5), p. 4018017.

Al-Washali, T., Sharma, S. and Kennedy, M. (2016) 'Methods of Assessment of Water Losses in Water Supply Systems: a Review', *Water Resource Management*, 30, pp. 4985–5001.

Al-Washili, T. *et al.* (2020) 'A Review of Non-Revenue Water Assessment Software Tools', *WIREs Water*, 7(2), p. 1413.

Alberizzi, J. C. *et al.* (2019) 'Speed and Pressure Controls of Pumps-as-Turbines Installed in Branch of Water-Distribution Network Subjected to Highly Variable Flow Rates', *Energies* 2019, Vol. 12, Page 4738, 12(24), p. 4738. doi: 10.3390/EN12244738.

Aldea, A. and Jernigan, W. (2016) 'Free software tools for water losses calculation', in *IWA Water Loss 2016*. Bangalore, India.

Alegre, H. *et al.* (2000) *Performance Indicators for Water Supply Systems*. 1st edn, *Manual of Best Practice Series*. 1st edn. London: IWA Publishing 'Manuals of Best Practice' series.

- Alegre, H. (2002) 'Performance Indicators as a Management Support Tool', in *Urban Water Supply Handbook*. Lisbon, Portugal: McGraw Hill, pp. 9.3–9.74.
- Alegre, H. et al. (2006) *Performance Indicators for Water Supply Services*. 2nd edn, *Manual of Best Practice Series*. 2nd edn. London: IWA Publishing 'Manual of Best Practice' series.
- Alex, J et al. (2018) 'Benchmark Simulation Model no. 1 (BSM1)'.
- Alkaseh, J. M. A. et al. (2013) 'Applying Minimum Night Flow to Estimate Water Loss Using Statistical Modeling: A Case Study in Kinta Valley, Malaysia', *Water Resource Management*, 27, pp. 1439–1455,.
- Alves Goulart, D. and Dutra Pereira, R. (2020) 'Autonomous pH control by reinforcement learning for electroplating industry wastewater', *Computers & Chemical Engineering*, 140, p. 106909. doi: 10.1016/J.COMPCHEMENG.2020.106909.
- Amoatey, P., Minke, R. and Steinmetz, H. (2018) 'Leakage estimation in developing country water networks based on water balance, minimum night flow and component analysis methods', *Water Practice and Technology*, 13(1), pp. 96–105,.
- Araujo, L. S., Ramos, H. and Coelho, S. T. (2006) 'Pressure Control for Leakage Minimisation in Water Distribution Systems Management', *Water Resources Management*, 20, pp. 133–149. doi: 10.1007/s11269-006-4635-3.
- Arbues, F., Garcia-Valinas, M. A. and Martinez-Espinera, R. (2003) 'Estimation of residential water demand: A state-of-the-art review', *Socio-Economics*, 32(1), pp. 81–102,.
- Arregui, F., Cabrera Jr., E. and Cobacho, R. (2007) *Integrated Water Meter Management*. Edited by F. Arregui, E. Cabrera Jr., and R. Cobacho. London: IWA Publishing.
- Arulkumaran, K. et al. (2017) 'Deep reinforcement learning: A brief survey', *IEEE Signal Processing Magazine*, 34(6), pp. 26–38. doi: 10.1109/MSP.2017.2743240.
- Aryal, S. K. et al. (2016) 'Assessing and Mitigating the Hydrological Impacts of Urbanisation in Semi-Urban Catchments Using the Storm Water Management Model', *Water Resources Management*, 30(14), pp. 5437–5454. doi: 10.1007/S11269-016-1499-Z.
- Assessment, U. E. N. C. for E. (2009) 'Pipe materials selection manual: water mains'.
- Babovic, V. et al. (2002) 'A data mining approach to modelling of water supply assets', *Urban Water*, 4(4), pp. 401–414. doi: 10.1016/S1462-0758(02)00034-1.

Bach, P. M. and Kodikara, J. K. (2017) 'Reliability of Infrared Thermography in Detecting Leaks in Buried Water Reticulation Pipes', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(9), pp. 4210–4224. doi: 10.1109/JSTARS.2017.2708817.

Baird, L. (1995) 'Residual Algorithms: Reinforcement Learning with Function Approximation', *Machine Learning Proceedings 1995*, pp. 30–37. doi: 10.1016/B978-1-55860-377-6.50013-X.

Barton, N A; Farewell, T. S.; Hallet, S. H.; Acland, T. F. (2019) 'Improving pipe failure predictions: Factors affecting pipe failure in drinking water networks', *Water Research*, 114926.

Battilotti, S., Sapienza, L. and Bertsekas, D. P. (2000) 'Dynamic Programming and Optimal Control , Part I', *Control*, 36, pp. 638–639. Available at: https://books.google.com/books/about/Dynamic_Programming_and_Optimal_Control.html?id=RjGQQgAACAAJ (Accessed: 7 April 2023).

Beattie, C.; Leibo, J. Z.; Teplyashin, D.; Ward, Tom; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; Schrittwieser, J.; Anderson, K; York, S.; Cant, M.; Cain, A; Bolton, A.; Gaffney, S.; King, H.; Hassabis, D.; Legg, S.; Petersen, S. (2016) 'DeepMind Lab'. Available at: <https://arxiv.org/abs/1612.03801v2> (Accessed: 4 May 2023).

Bellemare, M. G., Dabney, W. and Munos, R. (2017) 'A Distributional Perspective on Reinforcement Learning', *34th International Conference on Machine Learning, ICML 2017*, 1, pp. 693–711. Available at: <https://arxiv.org/abs/1707.06887v1> (Accessed: 10 May 2023).

Bellemare, M. G., Veness, J. and Bowling, M. (2013) 'The Arcade Learning Environment: An Evaluation Platform for General Agents', *Journal of Artificial Intelligence Research*, 47, pp. 253–279. Available at: <http://stella.sourceforge.net/> (Accessed: 14 February 2023).

Bellman, R. (1952) 'On the Theory of Dynamic Programming', *Proceedings of the National Academy of Sciences*, 38(8), pp. 716–719. doi: 10.1073/PNAS.38.8.716/ASSET/BADDE5C3-CE28-4677-B095-95015576EEBC/ASSETS/PNAS.38.8.716.FP.PNG.

Benjamin, M. M. (2014) *Water chemistry*.

Bertetto, A. M. and Ruggiu, M. (2001) 'In-pipe inch-worm pneumatic flexible robot', *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, 2, pp. 1226–1231. doi: 10.1109/AIM.2001.936886.

Bertsekas, D. P., Tsitsiklis, J. N. and Τσιτουκλής, Γ. N. (1996) 'Neuro-dynamic programming', p. 491.

- Bhagat, S. K.; Tiyasham W. W.; Resfay, O.; Tung, T. M.; Al-Ansari, N.; Salih, S. Q.; Yaseen, Z. M. (2019) 'Evaluating Physical and Fiscal Water Leakage in Water Distribution System', *Water*, 11, p. 2091.
- Bickerstaff, R.; Vaughn, M.; Stoker, G.; Hassard, M.; Garrett, M. (2002) 'Review of sensor technologies for in-line inspection of natural gas pipelines', *Sandia National Laboratories, Albuquerque, NM*.
- Bilal and Pant, M. (2022) 'Differential Evolution for Water Management Problems', *Studies in Computational Intelligence*, 1009, pp. 197–214. doi: 10.1007/978-981-16-8082-3_7/COVER.
- Bloembergen, D.; Tuyls, K.; Hennes, D.; Kaisers, M. (2015) 'Evolutionary dynamics of multi-agent learning: A survey', *Journal of Artificial Intelligence Research*. doi: 10.1613/jair.4818.
- Boaz, L., Kaijage, S. and Sinde, R. (2014) 'An overview of pipeline leak detection and location systems', *Proceedings of the 2nd Pan African International Conference on Science, Computing and Telecommunications, PACT 2014*, pp. 133–137. doi: 10.1109/SCAT.2014.7055147.
- Bonthuys, G. J., van Dijk, M. and Cavazzini, G. (2020) 'The optimization of energy recovery device sizes and locations in municipal water distribution systems during extended-period simulation', *Water (Switzerland)*, 12(9). doi: 10.3390/w12092447.
- Bowes, B. D.; Tavakoli, A.; Wang, C.; Heydarian, A.; Behl, M.; Beling, P.; Goodall, J. L. (2021) 'Flood mitigation in coastal urban catchments using real-time stormwater infrastructure control and reinforcement learning', *Journal of Hydroinformatics*, 23(3), pp. 529–547. doi: 10.2166/HYDRO.2020.080.
- Bradbeer, R.; Harrold, S.; Nickols, F.; Yeung, L. F. (1997) 'Underwater robot for pipe inspection', *Proceedings of the Annual Conference on Mechatronics and Machine Vision in Practice, MViP*, pp. 152–156. doi: 10.1109/MMVIP.1997.625313.
- Bruaset, S. and Sægrov, S. (2018) 'An Analysis of the Potential Impact of Climate Change on the Structural Reliability of Drinking Water Pipes in Cold Climate Regions', *Water 2018, Vol. 10, Page 411*, 10(4), p. 411. doi: 10.3390/W10040411.
- Bullock, J.; Luccioni, A.; Pham, K H.; Lam, C. S. N.; Luengo-Oroz, M. (2020) 'Mapping the landscape of Artificial Intelligence applications against COVID-19', *Journal of Artificial Intelligence Research*, 69, pp. 807–845. doi: 10.1613/JAIR.1.12162.
- Burn, S., Davis, P. and Schiller, T. (2005) 'Long-term performance prediction for PVC pipes'. Available at: <https://espace.library.uq.edu.au/view/UQ:193983> (Accessed: 21 January 2023).

- Buşoniu, L., Babuška, R. and De Schutter, B. (2008) 'A comprehensive survey of multiagent reinforcement learning', *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 38(2), pp. 156–172. doi: 10.1109/TSMCC.2007.913919.
- Campbell, M., Hoane, A. J. and Hsu, F. H. (2002) 'Deep Blue', *Artificial Intelligence*, 134(1–2). doi: 10.1016/S0004-3702(01)00129-1.
- Casillas, M. V., Garza-Castanon, L. E. and Puig, V. (2013) 'Extended-horizon analysis of pressure sensitivities for leak detection in water distribution networks: Application to the Barcelona network', *2013 European Control Conference, ECC 2013*, pp. 404–409. doi: 10.23919/ECC.2013.6669568.
- Cataldo, A.; Persico, R.; Leucci, G.; De Benedetto, E.; Cannazza, G.; Matera, L.; De Giorgi, L. (2014) 'Time domain reflectometry, ground penetrating radar and electrical resistivity tomography: A comparative analysis of alternative approaches for leak detection in underground pipes', *NDT & E International*, 62, pp. 14–28. doi: 10.1016/J.NDTEINT.2013.10.007.
- Chan, T. K., Chin, C. S. and Zhong, X. (2018) 'Review of Current Technologies and Proposed Intelligent Methodologies for Water Distributed Network Leakage Detection', *IEEE Access*, 6, pp. 78846–78867. doi: 10.1109/access.2018.2885444.
- Chang, A. (2019) 'Precision intensive care: A real-time artificial intelligence strategy for the future', *Pediatric Critical Care Medicine*, 20(2), pp. 194–195. doi: 10.1097/PCC.0000000000001883.
- Chen, K.; Wang, H.; Valverde-Pérez, B.; Zhai, S.; Vezzano, L.; Wang, A. (2021) 'Optimal control towards sustainable wastewater treatment plants based on multi-agent reinforcement learning', *Chemosphere*, 279, p. 130498. doi: 10.1016/J.CHEMOSPHERE.2021.130498.
- Choi, C., Jung, S. and Kim, S. (2004) 'Feeder pipe inspection robot with an inch-worm mechanism using pneumatic actuators', *International Journal of Control, Automation and Systems*, 4(1), pp. 87–95. doi: 10.1109/ROBIO.2004.1521902.
- Crini, G. and Lichtfouse, E. (2019) 'Advantages and disadvantages of techniques used for wastewater treatment', *Environmental Chemistry Letters*, 17, pp. 145–155. doi: 10.1007/s10311-018-0785-9.
- Croll, H. C.; Ikuma, K.; Ong, S. K.; Sarkar, S. (2023) 'Reinforcement learning applied to wastewater treatment process control optimization: Approaches, challenges, and path

forward', *Critical Reviews in Environmental Science and Technology*, 53(20), pp. 1775–1794. doi: 10.1080/10643389.2023.2183699.

Dabney, W. Rowland, M.; Bellemare, M. G.; Munos, R. (2017) 'Distributional Reinforcement Learning with Quantile Regression', *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 2892–2901. doi: 10.1609/aaai.v32i1.11791.

Dabney, W.; Ostrovski, G.; Silver, D.; Munos, R. (2018) 'Implicit Quantile Networks for Distributional Reinforcement Learning', *35th International Conference on Machine Learning, ICML 2018*, 3, pp. 1774–1787. Available at: <https://arxiv.org/abs/1806.06923v1> (Accessed: 10 May 2023).

Dabney, W.; Kurth-Nelson, Z.; Uchida, N.; Starkweather, C. K.; Hassabis, D.; Munos, R.; Botvinick, M. (2020) 'A distributional code for value in dopamine-based reinforcement learning', *Nature* 2020 577:7792, 577(7792), pp. 671–675. doi: 10.1038/s41586-019-1924-6.

Dai, P. D. and Li, P. (2014) 'Optimal Localization of Pressure Reducing Valves in Water Distribution Systems by a Reformulation Approach', *Water Resources Management*, 28(10). doi: 10.1007/s11269-014-0655-6.

Darweesh, M. S. (2022) 'Predicting the head leakage behaviour of cracks in pipe elbows', *Water SA*, 48(1), pp. 56–61. doi: 10.17159/wsa/2022.v48.i1.3910.

Datamatic Ltd. (2008) *Permalog+ - Automatic Meter Reading System - Leak Noise ...* Available at: <https://www.environmental-expert.com/products/permalog-leak-noise-loggers-122138> (Accessed: 6 September 2021).

Davila, M.; Davila Delgado, J. M.; Brilakis, I.; Middleton, C. (2016) 'Distributed monitoring of buried pipelines with Brillouin fiber optic sensors'. doi: 10.1680/TFITS1.61279.033.

Delipetrev, B., Jonoski, A. and Solomatine, D. P. (2017) 'A novel nested stochastic dynamic programming (nSDP) and nested reinforcement learning (nRL) algorithm for multipurpose reservoir optimization', *Journal of Hydroinformatics*, 19(1), pp. 47–61. doi: 10.2166/HYDRO.2016.243.

Demirci, S.; Yigit, E.; Eskidemir, I. H.; Ozdemir, C. (2012) 'Ground penetrating radar imaging of water leaks from buried pipes based on back-projection method', *NDT & E International*, 47, pp. 35–42. doi: 10.1016/J.NDTEINT.2011.12.008.

Department for Energy Security and Net Zero (2023) *Greenhouse gas reporting: conversion factors 2023 - GOV.UK*. Available at:

<https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2023> (Accessed: 17 October 2023).

Desharnais, J.; Gupta, V; Jagadeesan, R.; Panangaden, P. (2004) 'Metrics for labelled Markov processes', *Theoretical Computer Science*, 318(3), pp. 323–354. doi: 10.1016/J.TCS.2003.09.013.

Desnoyers, J. M. ; and Mcdonald, R. ; (no date) 'Failure modes and mechanisms in gray cast iron pipe NRCC-44218'. Available at: www.nrc.ca/irc/ircpubs (Accessed: 19 January 2023).

Dingus, M., Haven, J. and Austin, R. (2002) *Nondestructive, noninvasive assessment of underground pipelines*. Available at: https://books.google.com/books?hl=en&lr=&id=F8A_1r6JOHEC&oi=fnd&pg=PR7&ots=Qj_8QZE7mf&sig=Q3wC4GpAEhJ9gBH85LpYK6Y6GA (Accessed: 21 January 2023).

Dreyfus, S. E. and Law, A. M. (1977) 'The art and theory of dynamic programming, Volume 130', p. 284.

Duan, Y.; Chen, X.; Edu, C. X.; Schulman, J.; Abbeel, P.; Edu, P. (2016) 'Benchmarking Deep Reinforcement Learning for Continuous Control'. PMLR, pp. 1329–1338. Available at: <https://proceedings.mlr.press/v48/duan16.html> (Accessed: 4 May 2023).

Eck, B. J. and Mevissen, M. (2012) 'Valve Placement in Water Networks: Mixed-Integer Non-Linear Optimization with Quadratic Pipe Friction Valve Placement in Water Networks: Mixed-Integer Non-Linear Optimization with Quadratic Pipe Friction', *IBM Research Report*, 25307.

El-Abbasy, M. S.; Mosleh, F.; Senouci, A.; Zayed, T.; Al-Derham, H. (2016) 'Locating Leaks in Water Mains Using Noise Loggers', *Journal of Infrastructure Systems*, 22(3), p. 04016012. doi: 10.1061/(asce)is.1943-555x.0000305.

EL-Bagory, T. M. A. A. and Younan, M. Y. A. (2017) 'Crack Growth Behavior of Pipes Made from Polyvinyl Chloride Pipe Material', *Journal of Pressure Vessel Technology, Transactions of the ASME*, 139(1). doi: 10.1115/1.4033124/473309.

El-Zahab, S.; Asaad, A.; Mohammed Abdelkader, E; Zayed, T. (2017) 'Collective thinking approach for improving leak detection systems', *Smart Water 2017 2:1*, 2(1), pp. 1–10. doi: 10.1186/S40713-017-0007-9.

El-Zahab, S. and Zayed, T. (2019) 'Leak detection in water distribution networks: an introductory overview', *Smart Water*, 4(1). doi: 10.1186/s40713-019-0017-x.

Epa, U. (2010) 'CONTROL AND MITIGATION OF DRINKING WATER LOSSES IN DISTRIBUTION SYSTEMS'.

ESRI (2023) *What is GIS? | Geographic Information System Mapping Technology*. Available at: <https://www.esri.com/en-us/what-is-gis/overview> (Accessed: 11 September 2023).

Etikala, B., Madhav, S. and Somagouni, S. G. (2022) 'Urban water systems: An overview', 6, pp. 1–19. doi: 10.1016/B978-0-323-91838-1.00016-6.

Fan, X., Zhang, X. and Yu, X. (2022) 'A graph convolution network-deep reinforcement learning model for resilient water distribution network repair decisions', *Computer-Aided Civil and Infrastructure Engineering*, 37(12), pp. 1547–1565. doi: 10.1111/MICE.12813.

Farah, E. and Shahrour, I. (2017) 'Leakage detection using smart water system: combination of water balance and Automated Minimum Night Flow', *Water Resource Management*, 31(15), pp. 4821–4833,.

Farewell, T. S.; Hallett, S. H.; Hannam, J. A.; Jones, R. J. A. (2012) 'Infrastructure Transitions Research Consortium Working paper series Soil impacts on national infrastructure in the UK'.

Farewell, T. S.; Jude, S. and Pritchard, O. (2018) 'How the impacts of burst water mains are influenced by soil sand content', *Natural Hazards and Earth System Sciences*, 18(11), pp. 2951–2968. doi: 10.5194/NHESS-18-2951-2018.

Farley, B., Mounce, S. R. and Boxall, J. B. (2010) 'Field testing of an optimal sensor placement methodology for event detection in an urban water distribution network', *Urban Water Journal*, 7(6), pp. 345–356. doi: 10.1080/1573062X.2010.526230.

Farley, M.; Wyeth, G.; Ghazali, C. B. M.; Istander, A.; Singh, S. (2008) *The Manager's Non-Revenue Water Handbook*. Edited by N. V. Dijk, V. Raksakulthai, and E. Kirkwood. Ranhill Utilities and United States Agency for International Development (USAID).

Farley, M. and Trow, S. (2015) 'Losses in Water Distribution Networks: A Practitioners' Guide to Assessment, Monitoring and Control', *Water Intelligence Online*, 4(0). doi: 10.2166/9781780402642.

Ferrandez-Gamot, L.; Busson, P.; Blesa, J.; Tornil-Sin, S.; Puig, V.; Duviella, E.; Soldevila, A. (2015) 'Leak Localization in Water Distribution Networks using Pressure Residuals and Classifiers', *IFAC-PapersOnLine*, 48(21), pp. 220–225. doi: 10.1016/J.IFACOL.2015.09.531.

Filipe, J. *et al.* (2019) 'Data-driven predictive energy optimization in a wastewater pumping

station'. doi: 10.1016/j.apenergy.2019.113423.

Finn, C., Levine, S. and Abbeel, P. (2016) 'Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization', *33rd International Conference on Machine Learning, ICML 2016*, 1, pp. 95–107. Available at: <https://arxiv.org/abs/1603.00448v3> (Accessed: 10 May 2023).

Folkman, S. (2018) 'Water Main Break Rates In the USA and Canada: A Comprehensive Study Overall Pipe Breaks Up 27% In Six Years'.

Foundation, W. R. (2014) *Leakage Repair Data Collection Guide*. Available at: <https://www.waterrf.org/resource/leak-repair-data-collection-guide>. (Accessed: 4 June 2021).

Fu, G.; Jin, Y.; Sun, S.; Yuan, Z.; Butler, D. (2022) 'The role of deep learning in urban water management: A critical review', *Water Research*, 223. doi: 10.1016/j.watres.2022.118973.

Gao, Y.; Brennan, M. J.; Liu, Y.; Almeida, F. C.L.; Joseph, P. F. (2017) 'Improving the shape of the cross-correlation function for leak detection in a plastic water distribution pipe using acoustic signals', *Applied Acoustics*, 127, pp. 24–33. doi: 10.1016/J.APACOUST.2017.05.033.

García, V. J. and Cabrera, E. (2007) 'The minimum night flow method revisited', *8th Annual Water Distribution Systems Analysis Symposium 2006*, p. 35. doi: 10.1061/40941(247)35.

Geiger, G. (2006) 'State-of-the-Art in Leak Detection and Localisation', in *Pipeline Technology*. Hannover, Germany, pp. 1–25.

Geng, Z.; Hu, X.; Han, Y.; Zhong, Y. (2018) 'A Novel Leakage-Detection Method Based on Sensitivity Matrix of Pipe Flow: Case Study of Water Distribution Systems'. doi: 10.1061/(ASCE)WR.1943-5452.0001025.

Gomez, F. and Schmidhuber, J. (2005) 'Evolving modular fast-weight networks for control', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3697 LNCS, pp. 383–389. doi: 10.1007/11550907_61/COVER.

Gordon, G. J. (1995) 'Stable Function Approximation in Dynamic Programming', *Machine Learning Proceedings 1995*, pp. 261–268. doi: 10.1016/B978-1-55860-377-6.50040-2.

Gould, S. J. F.; Davis, P.; Beale, D. J.; Marlow, D. R. (2013) 'Failure analysis of a PVC sewer pipeline by fractography and materials characterization', *Engineering Failure Analysis*, 34, pp. 41–50. doi: 10.1016/J.ENGFAILANAL.2013.07.009.

Gross, W. *et al.* (1999); Hierl, T.; Scheuerpflug, H.; Schirl, U.; Schulz, M. J. 'Quality control of

heat pipelines and sleeve joints by infrared measurements', *Thermosense XXI*, 3700, pp. 63–69. doi: 10.1117/12.342275.

Gu, S. Lillicrap, T.; Sutskever, U.; Levine, S. (2016) 'Continuous Deep Q-Learning with Model-based Acceleration', *33rd International Conference on Machine Learning, ICML 2016*, 6, pp. 4135–4148. Available at: <https://arxiv.org/abs/1603.00748v1> (Accessed: 1 April 2023).

Guan, L. *et al.* (2019) 'A Review on Small-Diameter Pipeline Inspection Gauge Localization Techniques: Problems, Methods and Challenges', in *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1–6. doi: 10.1109/ICCSPA.2019.8713703.

Gullotta, A.; Campisano, A.; Creaco, E.; Modica, C. (2021) 'A Simplified Methodology for Optimal Location and Setting of Valves to Improve Equity in Intermittent Water Distribution Systems', *Water Resources Management*, 35(13), pp. 4477–4494. doi: 10.1007/S11269-021-02962-9/FIGURES/6.

Guoquan, W. and Yaoting, C. (1991) 'Test methods for gelation of PVC plastisol', *Polymer Testing*, 10(4), pp. 315–324. doi: 10.1016/0142-9418(91)90025-S.

Gupta, A.; Bokde, N.; Marathe, D.; Kulat, K. (2017) 'Leakage Reduction in Water Distribution Systems with Efficient Placement and Control of Pressure Reducing Valves Using Soft Computing Techniques', *Engineering, Technology & Applied Science Research*, 7(2). doi: 10.48084/etasr.1032.

Gupta, A. and Kulat, K. D. (2018) 'A Selective Literature Review on Leak Management Techniques for Water Distribution System', *Water Resources Management*. Springer Netherlands, pp. 3247–3269. doi: 10.1007/s11269-018-1985-6.

Hajgató, G., Paál, G. and Gyires-Tóth, B. (2020) 'Deep Reinforcement Learning for Real-Time Optimization of Pumps in Water Distribution Systems', *Journal of Water Resources Planning and Management*, 146(11). doi: 10.1061/(asce)wr.1943-5452.0001287.

Hamilton, S. (2009) 'ALC in Low Pressure Areas - It can be done', in *5th IWA Water Loss Reduction*. Cape Town.

Hamilton, S. and Charalambous, B. (2013) 'Leak Detection: Technology and Implementation', *Water Intelligence Online*, 12. doi: 10.2166/9781780404714.

Hamilton, S. and McKenzie, R. (2014) 'Meter logging and recording', in *Water Management and Water Loss*. London: IWA Publishing, pp. 87–107.

Hasan, M. M.; Lwin, K.; Imani, M.; Shabut, A.; Bittencourt, L. F.; Hossain, M. A. (2019) 'Dynamic multi-objective optimisation using deep reinforcement learning: benchmark, algorithm and an application to identify vulnerable zones based on water quality', *Engineering Applications of Artificial Intelligence*, 86, pp. 107–135. doi: 10.1016/J.ENGAPPAI.2019.08.014.

Van Hasselt, H., Guez, A. and Silver, D. (2016) 'Deep Reinforcement Learning with Double Q-learning', in *30th AAAI Conference on Artificial Intelligence*, pp. 2094–2100. Available at: www.aaai.org (Accessed: 4 May 2023).

Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Tassa, Y.; Erez, T. (2015) 'Learning Continuous Control Policies by Stochastic Value Gradients', *Advances in Neural Information Processing Systems*, 2015-January, pp. 2944–2952. Available at: <https://arxiv.org/abs/1510.09142v1> (Accessed: 9 May 2023).

Hernández-del-Olmo, F.; Gaudioso, E.; Dormido, R.; Duro, N. (2018) 'Tackling the start-up of a reinforcement learning agent for the control of wastewater treatment plants', *Knowledge-Based Systems*, 144, pp. 9–15. doi: 10.1016/J.KNOSYS.2017.12.019.

Hernández-del-Olmo, F.; Gaudioso, E.; Dormido, R.; Duro, N. (2016) 'Energy and Environmental Efficiency for the N-Ammonia Removal Process in Wastewater Treatment Plants by Means of Reinforcement Learning', *Energies* 2016, Vol. 9, Page 755, 9(9), p. 755. doi: 10.3390/EN9090755.

Hernandez-Leal, P., Kartal, B. and Taylor, M. E. (2018) 'A Survey and Critique of Multiagent Deep Reinforcement Learning'. doi: 10.1007/s10458-019-09421-1.

Hernandez-Leal, P., Kartal, B. and Taylor, M. E. (2019) 'Is multiagent deep reinforcement learning the answer or the question? A brief survey', *Autonomous Agents and Multi-Agent Systems*, 33(6).

Hindi, K. S. and Hamam, Y. M. (2007) 'LOCATING PRESSURE CONTROL ELEMENTS FOR LEAKAGE MINIMIZATION IN WATER SUPPLY NETWORKS: AN OPTIMIZATION MODEL', <http://dx.doi.org/10.1080/03052159108941076>, 17(4), pp. 281–291. doi: 10.1080/03052159108941076.

Ho, J. and Ermon, S. (2016) 'Generative Adversarial Imitation Learning', *Advances in Neural Information Processing Systems*, pp. 4572–4580. Available at: <https://arxiv.org/abs/1606.03476v1> (Accessed: 10 May 2023).

Hu, S.; Gao, J.; Zhong, D.; Wu, R.; Liu, L. (2023) 'Real-Time Scheduling of Pumps in Water

Distribution Systems Based on Exploration-Enhanced Deep Reinforcement Learning', *Systems* 2023, Vol. 11, Page 56, 11(2), p. 56. doi: 10.3390/SYSTEMS11020056.

Hu, Y. and Hubble, D. W. (2011) 'Factors contributing to the failure of asbestos cement water mains', <https://doi.org/10.1139/l06-162>, 34(5), pp. 608–621. doi: 10.1139/L06-162.

Huang, Y.; Fipps, G.; Maas, S. J.; Fletcher, R. S. (2010) 'Airborne remote sensing for detection of irrigation canal leakage', *Irrigation and Drainage*, 59(5), pp. 524–534. doi: 10.1002/IRD.511.

Hunaidi, O.; Chu, W.; Wang, A.; Guan, W. (2000) 'Detecting leaks in plastic pipes', *Journal - American Water Works Association*, 92(2), pp. 82–94. doi: 10.1002/J.1551-8833.2000.TB08819.X.

Hussain, A.; Kumari, R.; Sachan, S. G.; Sachan, A. (2021) 'Biological wastewater treatment technology: Advancement and drawbacks', *Microbial Ecology of Wastewater Treatment Plants*, pp. 175–192. doi: 10.1016/B978-0-12-822503-5.00002-3.

Hutsebaut-Buysse, M., Mets, K. and Latré, S. (2022) 'Hierarchical Reinforcement Learning: A Survey and Open Research Challenges', *Machine Learning and Knowledge Extraction* 2022, Vol. 4, Pages 172-221, 4(1), pp. 172–221. doi: 10.3390/MAKE4010009.

ILMSS Ltd. (2015) 'CheckCalcs V6b'.

Inaudi, D. and Glisic, B. (2008) 'Long-Range Pipeline Monitoring by Distributed Fiber Optic Sensing', *Proceedings of the Biennial International Pipeline Conference, IPC*, 3 PART B, pp. 763–772. doi: 10.1115/IPC2006-10287.

Instituto Tecnológico del Agua-Valencia Polytechnic University (2021) *Sigma Features*, ITA. Available at: <https://sigmalite.com/caracteristicas-en.php>. (Accessed: 22 June 2021).

Ishido, Y. and Takahashi, S. (2014) 'A New Indicator for Real-time Leak Detection in Water Distribution Networks: Design and Simulation Validation', *Procedia Engineering*, 89, pp. 411–417. doi: 10.1016/J.PROENG.2014.11.206.

Iskander, M. (2018) 'Geotechnical Underground Sensing and Monitoring', *Underground Sensing: Monitoring and Hazard Detection for Environment and Infrastructure*, pp. 141–202. doi: 10.1016/B978-0-12-803139-1.00003-5.

Ismail, I. N.; Anuar, AS.; Sahari, K. S. M. (2012) 'Developments of In-Pipe Inspection Robot: A Review', in *IEE Conference on Sustainable Utilization and Development in ENgineering and Technology (STUDENT)*. Journal of Mechanics of Continua and Mathematical Sciences, pp. 310–

315. doi: 10.26782/JMCMS.2020.05.00022.

Jacobsz, S. W. and Jahnke, S. I. (2019) 'Leak detection on water pipelines in unsaturated ground by discrete fibre optic sensing', <https://doi.org/10.1177/1475921719881979>, 19(4), pp. 1219–1236. doi: 10.1177/1475921719881979.

Jefferson, A. J.; Bhaskar, A. S.; Hopkins, K. G.; Fanelli, R.; Avellaneda, P. M.; McMillan, S. K. (2017) 'Stormwater management network effectiveness and implications for urban watershed function: A critical review', *Hydrological Processes*, 31(23), pp. 4056–4080. doi: 10.1002/HYP.11347.

Jekel, M. (1996) 'Water Supply. Von A. C. Twort, F. M. Law, F. W. Crowley, D. D. Ratnayaka 4. Auflage. Edward Arnold, London, 1994. ISBN 0–340–57587–5, 511 S., £ 39,50', *Acta Hydrochimica et Hydrobiologica*, 24(5). doi: 10.1002/aheh.19960240517.

Jiang, J.-Q. (2015) 'The role of coagulation in water treatment This review comes from a themed issue on Separation engineering', *Current Opinion in Chemical Engineering*, 8, pp. 36–44. doi: 10.1016/j.coche.2015.01.008.

Jotte, L., Raspati, G. and Azrague, K. (2017) *REVIEW OF STORMWATER MANAGEMENT PRACTICES*. Available at: www.klima2050.no (Accessed: 26 September 2023).

Joung, O. J. and Kim, Y. H. (2006) 'Application of an IR thermographic device for the detection of a simulated defect in a pipe', *Sensors*, 6(10), pp. 1199–1208. doi: 10.3390/S6101199.

Jowitt, P. W. and Xu, C. (1990) 'Optimal Valve Control in Water Distribution Networks', *Journal of Water Resources Planning and Management*, 116(4), pp. 455–472. doi: 10.1061/(ASCE)0733-9496(1990)116:4(455).

Jung, D.; Kang, D.; Liu, J.; Lansey, K. (2015) 'Improving the rapidity of responses to pipe burst in water distribution systems: a comparison of statistical process control methods', *Journal of Hydroinformatics*, 17(2), pp. 307–328. doi: 10.2166/HYDRO.2014.101.

Kadri, A., Yaacoub, E. and Mushtaha, M. (2014) 'Empirical evaluation of acoustical signals for leakage detection in underground plastic pipes', *Proceedings of the Mediterranean Electrotechnical Conference - MELECON*, pp. 54–58. doi: 10.1109/MELCON.2014.6820506.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; Levine, S. (2018) 'QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation'. Available at: <https://arxiv.org/abs/1806.10293v3> (Accessed: 26 March 2023).

- Kanakoudis, V.; Tsitsifli, S.; Zouboulis, A.; Samaras, P.; Doufas, A.; Banovec, P. (2011) 'Integration the WB/PI Calc-UTH Water Audit Tool Based on a Modification of the IWA Standard Water Balance, Into a DSS for NRW reduction Strategies Prioritization', in *4th BWA & EWRA International Conference on Water Loss Reduction in Water Supply Systems*. Sofia, Bulgaria.
- Kanakoudis, V. and Tsitsifli, S. (2010) 'Water volume vs. revenues oriented water balance calculation for urban water networks: the "Minimum Charge Difference" component makes a difference!', in *IWA International Conference 'WaterLoss'*. Sao Paolo, Brazil.
- Kentish, S. and Stevens, G. (2001) 'Innovations in separations technology for the recycling and re-use of liquid waste streams', *Chemical Engineering Journal*, 84(2). Available at: <https://www.sciencedirect.com/science/article/pii/S1385894701001991> (Accessed: 25 September 2023).
- Khawandi, S., Daya, B. and Chauvet, P. (2010) 'Automated Monitoring System for Fall Detection in the Elderly', *International Journal of Image Processing (IJIP)*, (4), p. 476.
- Khulief, Y. A.; Khalifa, A.; Mansour, R. B.; Habib, M. A. (2012) 'Acoustic Detection of Leaks in Water Pipelines Using Measurements inside Pipe', *Journal of Pipeline Systems Engineering and Practice*, 3(2), pp. 47–54. doi: 10.1061/(ASCE)PS.1949-1204.0000089.
- Kingma, D. P. and Welling, M. (2013) 'Auto-Encoding Variational Bayes', *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. Available at: <https://arxiv.org/abs/1312.6114v11> (Accessed: 9 May 2023).
- Kirchner, F. and Hertzberg, J. (1997) 'A Prototype Study of an Autonomous Robot Platform for Sewerage System Maintenance', *Autonomous Robots 1997 4:4*, 4(4), pp. 319–331. doi: 10.1023/A:1008896121662.
- Kirkham, R. *et al.* (2016) 'PIRAT—A System for Quantitative Sewer Pipe Assessment', <http://dx.doi.org/10.1177/02783640022067959>, 19(11), pp. 1033–1053. doi: 10.1177/02783640022067959.
- Kiss, G., Konez, K. and Melinte, C. (2007) 'WaterPipe project: An innovative high resolution ground penetration imaging radar for detecting water pipes and for detecting leaks and a decision support system for the rehabilitation management of the water pipeline', in *IWA Water Loss Conference*. Bucharest, Romania, pp. 622–631.
- Kılıç, Ş. *et al.* (2023) 'Sustainable development of energy, water and environment systems in

the critical decade for climate action', *Energy Conversion and Management*, 296, p. 117644. doi: 10.1016/J.ENCONMAN.2023.117644.

Kohl, N. and Stone, P. (2004) 'Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion', pp. 2619–2624. Available at: <http://www.cs.utexas.edu/~%7Bnate,pstone%7D> (Accessed: 6 February 2023).

Koldzo, D. and Vucijak, B. (2013) 'Testing Innovative Software Tool CalculEAKator for Water Balance Evaluation and Water Loss Reduction in Tuzla Project', in *6th International water loss conference*. Sofia, Bulgaria.

Kolesnik, M., Behavior, H. S. (2002) 'Visual orientation and motion control of MAKRO-adaptation to the sewer environment', in Proc. Int. Conf. Simulation of Adaptive Behaviour. *Citeseer*. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.7924&rep=rep1&type=pdf> (Accessed: 24 August 2021).

Konda, V. R. and Tsitsiklis, J. N. (1999) 'ON ACTOR-CRITIC ALGORITHMS', *Advances in Neural Information Processing Systems*, 42(4), pp. 1143–1166. doi: 10.1137/S0363012901385691.

Kool, W., Van Hoof, H. and Welling, M. (2018) 'Attention, Learn to Solve Routing Problems!', *7th International Conference on Learning Representations, ICLR 2019*. doi: 10.48550/arxiv.1803.08475.

Koutník, J.; Cuccu, G.; Schmidhuber, J.; Gomez, F. (2013) 'Evolving Large-Scale Neural Networks for Vision-Based Reinforcement Learning', in *Genetic and Evolutionary Computation Conference*. Available at: <http://www.idsia.ch/~koutnik/images/octo> (Accessed: 26 March 2023).

Kumar, V. and Kanal, L. N. (1988) 'The CDP: A unifying formulation for heuristic search, dynamic programming, and branch-and-bound', *Search in Artificial Intelligence*, pp. 1–27. doi: 10.1007/978-1-4613-8788-6_1.

Kuntze, H. B. and Haffner, H. (1998) 'Experiences with the development of a robot for smart multisensoric pipe inspection', *Proceedings - IEEE International Conference on Robotics and Automation*, 2, pp. 1773–1778. doi: 10.1109/ROBOT.1998.677423.

KVS (no date) *Tracer Gas Detection*. Available at: <http://www.leakdetection-technology.com/science/leak-detection-with-tracer-gas-methods.html> (Accessed: 10 September 2021).

Lai, T. L. and Robbins, H. (1985) 'Asymptotically efficient adaptive allocation rules', *Advances in Applied Mathematics*, 6(1). doi: 10.1016/0196-8858(85)90002-8.

Lambert, A. (1994) 'Accounting for Losses: The Bursts and Background Concept', *Water and Environment*, 8, pp. 205–214,.

Lambert, A.; Brown, T.; Takizawa, M.; Weimer, D. (1999) 'A review of performance indicators for real losses from water supply systems', *Water Supply*, 48(6), pp. 227–27,.

Lambert, A. (2001) 'What do we know about pressure-leakage relationships in distribution systems?', in *IWA conference in Systems approach to leakage control and water distribution system management*.

Lambert, A.; Charalambous, B.; Fantozzi, M.; Kovac, J.; Rizzo, A.; John, S. G. S. (2004) '14 Years Experience of using IWA Best Practice Water Balance and Water Loss Performance Indicators in Europe', in *IWA Specialized Conference: Water Loss*. Vienna, Italy.

Lambert, A. (2009) 'Ten years' experience in using the UARL formula to calculate infrastructure leakage index', in *IWA water loss conference*. Cape Town.

Lambert, A. (no date) *The LEAKSSuite Library is the next step in the LEAKSSuite story*. Leakssuite Library Ltd. Available at: <https://www.leakssuitelibrary.com/the-leakssuite-library-is-the-next-step-in-the-leakssuite-story> (Accessed: 10 June 2021).

Lapan, M. (2019) 'Deep Reinforcement Learning Learning Hands-on', *Reinforcement Learning for Cyber-Physical Systems*, pp. 125–154. Available at: <https://www.packtpub.com/product/deep-reinforcement-learning-hands-on-second-edition/9781838826994> (Accessed: 24 February 2023).

Lay-Ekuakille, A.; Vendramin, G.; Trotta, A.; Vanderbemden, P. (2009) 'STFT-BASED SPECTRAL ANALYSIS OF URBAN WATERWORKS LEAKAGE DETECTION'. Available at: <http://smaasis-misure.unile.it> (Accessed: 19 October 2021).

Lee, M. R. and Lee, J. H. (2000) 'Acoustic Emission Technique for Pipeline Leak Detection', *Key Engineering Materials*, 183–187(187 PART 2), pp. 887–892. doi: 10.4028/WWW.SCIENTIFIC.NET/KEM.183-187.887.

Lee, P. J.; Vítkovský, J. P.; Lambert, M. F.; Simpson, A. R.; Liggett, J. A. (2005) 'Frequency Domain Analysis for Detecting Pipeline Leaks', *Journal of Hydraulic Engineering*, 131(7), pp. 596–604. doi: 10.1061/(ASCE)0733-9429(2005)131:7(596).

- Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. (2016) 'End-to-end training of deep visuomotor policies', *Journal of Machine Learning Research*.
- Levine, S. and Van De Panne, M. (2018) 'DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills', *ACM Trans. Graph*, 37(143), p. 18. doi: 10.1145/3197517.3201311.
- Li, C.; Zheng, P.; Yin, Y.; Wang, B.; Wang, L. (2023) 'Deep reinforcement learning in smart manufacturing: A review and prospects', *CIRP Journal of Manufacturing Science and Technology*, 40, pp. 75–101. doi: 10.1016/J.CIRPJ.2022.11.003.
- Li, H.; Li, H.; Pei, H.; Li, Z. (2019) 'Leakage detection of HVAC pipeline network based on pressure signal diagnosis Article History', *Journal of Building Simulation*, 12(4). doi: 10.1007/s12273-019-0546-0.
- Li, Y. (2017) 'Deep Reinforcement Learning: An Overview'. doi: 10.48550/arxiv.1701.07274.
- Li, Z.; Bai, L.; Tian, W.; Yan, H.; Hu, W.; Xin, K.; Tao, T. (2023) 'Online Control of the Raw Water System of a High-Sediment River Based on Deep Reinforcement Learning', *Water* 2023, Vol. 15, Page 1131, 15(6), p. 1131. doi: 10.3390/W15061131.
- Libbrecht, M. W. and Noble, W. S. (2015) 'Machine learning applications in genetics and genomics', *Nature Reviews Genetics* 2015 16:6, 16(6), pp. 321–332. doi: 10.1038/nrg3920.
- Liberatore, S. and Sechi, G M (2009) 'Location and Calibration of Valves in Water Distribution Networks Using a Scatter-Search Meta-heuristic Approach', *Water Resour Manage*, 23, pp. 1479–1495. doi: 10.1007/s11269-008-9337-6.
- Liemberger & Partners (2020) *WB-EasyCalc v6.12*. Available at: <http://www.liemberger.cc/index.html>. (Accessed: 20 May 2021).
- Liemberger, R. and Farley, M. (2005) 'Developing a Non-Revenue Water Reduction Strategy Part 1: Investigating and Assessing Water Losses', in *IWA Specialised Conference: The 4th IWA World Water Congress*. Marakech, Morocco.
- Liljebck, P.; Pettersen, K. Y.; Stavadahl, O.; Gravadahl, J. T. (2012) 'A review on modelling, implementation, and control of snake robots', *Robotics and Autonomous Systems*, 60(1), pp. 29–40. doi: 10.1016/J.ROBOT.2011.08.010.
- Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. (2016) 'Continuous control with deep reinforcement learning', in *4th International Conference on*

Learning Representations, ICLR 2016 - Conference Track Proceedings.

Lim, J.; Park, H.; Moon, S.; Kim, B. (2007) 'Pneumatic robot based on inchworm motion for small diameter pipe inspection', *2007 IEEE International Conference on Robotics and Biomimetics, ROBIO*, pp. 330–335. doi: 10.1109/ROBIO.2007.4522183.

Lipps, W. C., Braun-Howland, E. B. and Baxter, T. E. (2022) 'Standard methods for the examination of water and wastewater', p. 1536.

Liu, X.-Y.; Xiong, Z.; Zhong, S.; Yang, H.; Walid, A. (2018) 'Practical Deep Reinforcement Learning Approach for Stock Trading'.

Liu, Z. and Kleiner, Y. (2013) 'State of the art review of inspection technologies for condition assessment of water pipes', *Measurement*, 46(1), pp. 1–15. doi: 10.1016/J.MEASUREMENT.2012.05.032.

Loubet, P.; Roux, P.; Loiseau, E.; Bellon-Maurel, V. (2014) 'Life cycle assessments of urban water systems: A comparative analysis of selected peer-reviewed literature'. doi: 10.1016/j.watres.2014.08.048.

Lowet, A. S.; Zheng, Q.; Matias, S.; Drugowitsch, J.; Uchida, N. (2020) 'Distributional Reinforcement Learning in the Brain', *Trends in Neurosciences*, 43(12), pp. 980–997. doi: 10.1016/j.tins.2020.09.004.

Luong, N. C.; Hoang, D. T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y. C.; Kim, D. (2019) 'Applications of Deep Reinforcement Learning in Communications and Networking: A Survey', *IEEE Communications Surveys and Tutorials*, 21(4), pp. 3133–3174. doi: 10.1109/COMST.2019.2916583.

Mace, M. (2020) *Water Industry Launches First Sector Wide Innovation Strategy*, *Water.org*. Available at: <https://www.water.org.uk/news-item/water-industry-launches-first-sector-wide-innovation-strategy> (Accessed: 29 December 2020).

Maier, H. R.; Kapelan, Z.; Kasprzyk, J.; Kollat, J.; Matott, L. S.; Cunha, M. C.; Dandy, G. C.; Gibbs, M. S.; Keedwell, E.; Marchi, A.; Ostfeld, A.; Savic, D.; Solomatine, D. P.; Vrugt, J. A.; Zecchin, A. C.; Minsker, B. S.; Barbour, E. J.; Kuczera, G.; Pasha, F.; Castelletti, A.; Giuliani, M.; Reed, P. M. (2014) 'Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions', *Environmental Modelling & Software*, 62, pp. 271–299. doi: 10.1016/J.ENVSOFT.2014.09.013.

Makropoulos, C. and Bouziotas, D. (2023) 'Artificial intelligence for decentralized water

systems: A smart planning agent based on reinforcement learning for off-grid camp water infrastructures', *Journal of Hydroinformatics*, 25(3), pp. 912–926. doi: 10.2166/HYDRO.2023.168.

Mala-Jetmarova, H., Sultanova, N. and Savic, D. (2018) 'Lost in optimisation of water distribution systems? A literature review of system design', *Water (Switzerland)*, 10(3). doi: 10.3390/W10030307.

Malik, H.; Srivastava, S.; Sood, Y. R.; Ahmad, A. (eds) (2019) 'Applications of Artificial Intelligence Techniques in Engineering', in. Singapore: Springer Singapore (Advances in Intelligent Systems and Computing). doi: 10.1007/978-981-13-1819-1.

Mania, H., Guy, A. and Recht, B. (2018) 'Simple random search provides a competitive approach to reinforcement learning'. Available at: <https://github.com/modestyachts/ARS>. (Accessed: 31 October 2023).

Martínez-Codina; Castillo, M.; González-Zeas, D.; Garrote, L. (2015) 'Pressure as a predictor of occurrence of pipe breaks in water distribution networks', <http://dx.doi.org/10.1080/1573062X.2015.1024687>, 13(7), pp. 676–686. doi: 10.1080/1573062X.2015.1024687.

May, J. (1994) 'Leakage, Pressure and Control', in *BICS International Conference on Leakage Control*. London, UK.

McDonnell, B.; Ratliff, K; Tryby, M.; Wu, J.; Mullapudi, A. (2020) 'PySWMM: The Python Interface to Stormwater Management Model (SWMM)', *Journal of Open Source Software*, 5(52), p. 2292. doi: 10.21105/JOSS.02292.

McKenzie, R. (1999) *Development of a standardised approach to evaluate burst and background losses in water distribution systems in South Africa - SanFlow user guide*. Pretoria, South africa: South African Water Research Commission.

McKenzie, R., Lambert, A.; Kock, J.; Mtshweni, W. (2002) *Development of a simple and pragmatic approach to benchmark real losses in potable water distribution systems - BenchLeak user guide*. Pretoria, South Africa: South African Water Research Commission.

McKenzie, R. (2007) *Aqualite: Water Balance Software User guide*. Pretoria, South Africa: South Africa Water Research Commission.

McKenzie, R. and Bhagwan, J. (2004) *Introduction to WRC Tools to Manage Non-Revenue Water*. Pretoria, South africa: South African Water Research Commission.

- McKenzie, R. and Lambert, A. (2002) *Development of a Windows based package for assessing appropriate levels of active leakage control in potable water distribution systems - EconoLeak user guide*. Pretoria, South Africa: South African Water Research Commission.
- McKenzie, R. and Lambert, A. (2008) *Benchmarking of Water Losses in New Zealand Manual*. New Zealand.
- McKenzie, R. and Langenhoven, S. (2001) *Development of a pragmatic approach to evaluate the potential savings from pressure management in potable water distribution systems - PresMac user guide*. Pretoria, South Africa: South African Water Research Commission.
- McKenzie, R., Meyer, N. and Lambert, A. (2002) *Calculating Hour-Day Factors for Potable Water Distribution Systems in South Africa - HDF User Guide*. Pretoria, South Africa: South African Water Research Commission.
- McKenzie, R. and Seago, C. (2008) 'Assessment of Real Losses in potable water distribution systems: some recent developments', *Water Science and Technology: Water Supply*, 5(1), pp. 33–40.
- Mehdi, D. and Asghar, A. (2019) 'Pressure Management of Large-Scale Water Distribution Network Using Optimal Location and Valve Setting', *Water Resources Management*, 33(14), pp. 4701–4713. doi: 10.1007/S11269-019-02381-X/FIGURES/10.
- Menciassi, A.; Park, Jong H.; Lee, S.; Gorini, S.; Dario, P.; Park, J. O. (2002) 'Robotic solutions and mechanisms for a semi-autonomous endoscope', *IEEE International Conference on Intelligent Robots and Systems*, 2, pp. 1379–1384. doi: 10.1109/IRDS.2002.1043947.
- Minsky, M. (1961) 'Steps Toward Artificial Intelligence', *Proceedings of the IRE*, 49(1), pp. 8–30. doi: 10.1109/JRPROC.1961.287775.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. (2013) 'Playing Atari with Deep Reinforcement Learning'. doi: 10.48550/arxiv.1312.5602.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; Hassabis, D. (2015) 'Human-level control through deep reinforcement learning', *Nature*, 518(7540). doi: 10.1038/nature14236.
- Mnih, V.; Badia, A. P.; Mirza, L.; Graves, A.; Harley, T.; Lillicrap, T. P.; Silver, D.; Kavukcuoglu, Koray. (2016) 'Asynchronous Methods for Deep Reinforcement Learning', *33rd International Conference on Machine Learning, ICML 2016*, 4, pp. 2850–2869. Available at:

<https://arxiv.org/abs/1602.01783v2> (Accessed: 1 April 2023).

Moraleda, J., Ollero, A. and Orte, M. (1999) 'A robotic system for internal inspection of water pipelines', *IEEE Robotics Automation Magazine*, 6(3), pp. 30–41. doi: 10.1109/100.793698.

Mosavi, A.; Faghan, Y.; Ghamisi, P.; Duan, P.; Ardabili, S. F.; Salwana, E; Band, S. S. (2020) 'Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics', *Mathematics 2020, Vol. 8, Page 1640*, 8(10), p. 1640. doi: 10.3390/MATH8101640.

Mosetlhe, T. C. Hamam, Y,; Du, S.; Monacelli, E. (2020) 'A survey of pressure control approaches in water supply systems', *Water (Switzerland)*. doi: 10.3390/W12061732.

Mounce, S.; Boxall, J. B.; Machell, J.; Mounce, S. R.; Boxall, J. B.; Machell, J. (2007) 'An Artificial Neural Network/Fuzzy Logic system for DMA flow meter data analysis providing burst identification and size estimation Transient Mobilisation of Adhered Particles in DWDS View project Assessing the Underworld View project An Artificial Neural Network/Fuzzy Logic system for DMA flow meter data analysis providing burst identification and size estimation'. Available at: <https://www.researchgate.net/publication/221936179> (Accessed: 30 September 2021).

Mounce, S. R.; Khan, A.; Wood, A. S.; Day, A. J.; Widdop, P. D.; Machell, J. (2003) 'Sensor-fusion of hydraulic data for burst detection and location in a treated water distribution system', *Information Fusion*, 4(3), pp. 217–229. doi: 10.1016/S1566-2535(03)00034-4.

Mounce, S. R., Boxall, J. B. and Machell, J. (2009) 'Development and Verification of an Online Artificial Intelligence System for Detection of Bursts and Other Abnormal Flows', *Journal of Water Resources Planning and Management*, 136(3), pp. 309–318. doi: 10.1061/(ASCE)WR.1943-5452.0000030.

Mounce, S. R. and Machell, J. (2007) 'Burst detection using hydraulic data from water distribution systems with artificial neural networks', <http://dx.doi.org/10.1080/15730620600578538>, 3(1), pp. 21–31. doi: 10.1080/15730620600578538.

Mounce, S. R., Mounce, R. B. and Boxall, J. B. (2011) 'Novelty detection for time series data analysis in water distribution systems using support vector machines', *Journal of Hydroinformatics*, 13(4), pp. 672–686. doi: 10.2166/HYDRO.2010.144.

Mullapudi, A.; Lewis, M. J.; Gruden, C. L.; Kerkez, B. (2020a) 'Deep reinforcement learning for the real time control of stormwater systems', *Advances in Water Resources*, 140. doi:

10.1016/j.advwatres.2020.103600.

Mutikanga, H. E., Sharma, S. and Vairavamorthy, K. (2011) 'Assessment of Apparent losses in Urban Water Systems', *Water and Environment*, 25(13), pp. 327–335,.

Nair, S.; George, B.; Malano, H. M.; Arora, M.; Nawarathna, B. (2014) 'Water–energy–greenhouse gas nexus of urban water systems: Review of concepts, state-of-art and methods', *Resources, Conservation and Recycling*, 89, pp. 1–10. doi: 10.1016/J.RESCONREC.2014.05.007.

Nam, K. J.; Heo, S. K.; Loy-Benitez, J.; Ifaei, P.; Yoo, C. K. (2020) 'An autonomous operational trajectory searching system for an economic and environmental membrane bioreactor plant using deep reinforcement learning', *Water Science and Technology*, 81(8), pp. 1578–1587. doi: 10.2166/WST.2020.053.

Nazari, M.; Oroojlooy, A.; Takáč, M.; Snyder, L. V. (2018) 'Reinforcement Learning for Solving the Vehicle Routing Problem', *Advances in Neural Information Processing Systems*, 2018-December, pp. 9839–9849. doi: 10.48550/arxiv.1802.04240.

Negm, A., Ma, X. and Aggidis, G. (2023a) 'Review of leakage detection in water distribution networks', *IOP Conference Series: Earth and Environmental Science*, 1136(1), p. 012052. doi: 10.1088/1755-1315/1136/1/012052.

Negm, A., Ma, X. and Aggidis, G. (2023b) 'Water Pressure Optimisation for Leakage Management Using Q Learning', *2023 IEEE Conference on Artificial Intelligence (CAI)*, pp. 270–271. doi: 10.1109/CAI54212.2023.00120.

Ng, A. Y. and Russell, S. (2000) 'Algorithms for Inverse Reinforcement Learning', in *International Conference of Machine learning*, pp. 663–670. Available at: http://www.eecs.harvard.edu/cs286r/courses/spring06/papers/ngruss_irl00.pdf (Accessed: 10 May 2023).

Nguyen, H. and La, H. (2019) 'Review of Deep Reinforcement Learning for Robot Manipulation', *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pp. 590–595. doi: 10.1109/IRC.2019.00120.

Nguyen, T. T., Nguyen, N. D. and Nahavandi, S. (2020) 'Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications', *IEEE Transactions on Cybernetics*, 50(9), pp. 3826–3839. doi: 10.1109/TCYB.2020.2977374.

Nichols, J. A., Herbert Chan, H. W. and Baker, M. A. B. (2019) 'Machine learning: applications of artificial intelligence to imaging and diagnosis', *Biophysical Reviews*, 11(1), pp. 111–118. doi:

10.1007/S12551-018-0449-9/METRICS.

Nicolini, M. (2011) 'Optimal pressure management in water networks: Increased efficiency and reduced energy costs', in *2011 Defense Science Research Conference and Expo, DSR 2011*. doi: 10.1109/DSR.2011.6026834.

Nicolini, M. and Zovatto, L. (2009) 'Optimal Location and Control of Pressure Reducing Valves in Water Networks', *Journal of Water Resources Planning and Management*, 135(3), pp. 178–187. doi: 10.1061/(ASCE)0733-9496(2009)135:3(178).

Task Committee on Water Pipeline Condition Assessment (2017) 'Water Pipeline Condition Assessment', *Water Pipeline Condition Assessment*, ASCE Publishing, pp. 1–206. doi: 10.1061/9780784414750.

OFWAT (2004) 'Updating the overall performance assessment (OPA)-Conclusions and methodology for 2004-05 onwards'.

OFWAT (2022) *Leakage in the water industry - Ofwat*. Available at: <https://www.ofwat.gov.uk/leakage-in-the-water-industry/> (Accessed: 25 July 2023).

Olsson, G. (2012) 'Water and Energy Nexus', *Life cycle assessment and water management-related issues. - (Quaderns de medi ambient ; 4)*, pp. 137–164. doi: 10.1400/241100.

Osband, I. ; Blundell, C.; Pritzel, A.; Van Roy, B. (2016) 'Deep exploration via bootstrapped DQN', in *Advances in Neural Information Processing Systems*.

Paine, T. Le; Colmenarejo, S. G.; Wang, Z.; Reed, S.; Aytar, Y.; Pfaff, T.; Hoffman, M. W.; Barth-Maron, G.; Cabi, S.; Budden, D.; De Freitas, N. (2018) 'One-Shot High-Fidelity Imitation: Training Large-Scale Deep Nets with RL'. Available at: <https://arxiv.org/abs/1810.05017v1> (Accessed: 10 May 2023).

Pang, J. W.; Yang, S. S.; He, L.; Chen, Y. D.; Cao, G. L.; Zhao, L.;

Wang, X. Y.; Ren, N. Q. (2019) 'An influent responsive control strategy with machine learning: Q-learning based optimization method for a biological phosphorus removal system', *Chemosphere*, 234, pp. 893–901. doi: 10.1016/J.CHEMOSPHERE.2019.06.103.

Panjapornpon, C.; Chinchalongporn, P.; Bardeeniz, S.; Makkayatorn, R.; Wongpunnawat, W. (2022) 'Reinforcement Learning Control with Deep Deterministic Policy Gradient Algorithm for Multivariable pH Process', *Processes 2022, Vol. 10, Page 2514*, 10(12), p. 2514. doi: 10.3390/PR10122514.

- Pathak, D.; Agrawal, P.; Efros, A. A.; Darrell, T. (2017) 'Curiosity-Driven Exploration by Self-Supervised Prediction', in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. doi: 10.1109/CVPRW.2017.70.
- Peng, X. B.; Kanazawa, A.; Toyer, S.; Abbeel, P.; Levine, S. (2018) 'Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow', *7th International Conference on Learning Representations, ICLR 2019*. Available at: <https://arxiv.org/abs/1810.00821v4> (Accessed: 10 May 2023).
- Perez, R.; Sanz, G.; Puig, V.; Quevedo, J.; Cuguero Escofet, M. A.; Nejjari, F.; Meseguer, J.; Cembrano, G.; Mirats Tur, J. M.; Sarrate, R. (2014) 'Leak localization in water networks: A model-based methodology using pressure sensors applied to a real network in barcelona [applications of control]', *IEEE Control Systems*, 34(4), pp. 24–36. doi: 10.1109/MCS.2014.2320336.
- Pérez, R.; Puig, V.; Pascual, J.; Quevedo, J.; Landeros, E.; Peralta, A. (2011) 'Methodology for leakage isolation using pressure sensitivity analysis in water distribution networks', *Control Engineering Practice*, 19(10), pp. 1157–1167. doi: 10.1016/J.CONENGPRAC.2011.06.004.
- Pham, D. D. (2018) 'Efficient Optimization of Pressure Regulation in Water Distribution Systems Using a New-Relaxed Pressure Reducing Valve Model', *Vietnam Journal of Science and Technology*, 56(4). doi: 10.15625/2525-2518/56/4/10571.
- Pleines, M.; Pallasch, M.; Zimmer, F.; Preuss, M. (2022) 'Generalization, Mayhems and Limits in Recurrent Proximal Policy Optimization'. Available at: <https://arxiv.org/abs/2205.11104v1> (Accessed: 27 December 2023).
- Pomerleau, D. A. (1989) 'ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK', *Advances in Neural Information Processing Systems 1*, 1, pp. 305–315.
- Pozos-Estrada, O.; Sánchez-Huerta, A.; Breña-Naranjo, J. A.; Pedrozo-Acuña, A. (2016) 'Failure Analysis of a Water Supply Pumping Pipeline System', *Water* 2016, Vol. 8, Page 395, 8(9), p. 395. doi: 10.3390/W8090395.
- Price, E., Abhijith, G. R. and Ostfeld, A. (2022) 'Pressure management in water distribution systems through PRVs optimal placement and settings', *Water Research*, 226, p. 119236. doi: 10.1016/J.WATRES.2022.119236.
- Pritchard, O. G., Hallett, S. H. and Farewell, T. S. (2015) 'Soil geohazard mapping for improved asset management of UK local roads', *Natural Hazards and Earth System Sciences*, 15(9), pp.

2079–2090. doi: 10.5194/NHESS-15-2079-2015.

Pritchard, Oliver G., Hallett, S. H. and Farewell, T. S. (2015) 'Soil impacts on UK infrastructure: current and future climate', *https://doi.org/10.1680/ensu.13.00035*, 167(4), pp. 170–184. doi: 10.1680/ENSU.13.00035.

Prudencio, R. F., Maximo, M. R. O. A. and Colombini, E. L. (2022) 'A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems'. doi: 10.1109/TNNLS.2023.3250269.

Pure Technologies (no date) *SmartBall - Leak and Gas Pocket Detection*. Available at: <https://puretechltd.com/technology/smartball-leak-detection/> (Accessed: 29 March 2022).

Puterman, M. L. (1990) 'Chapter 8 Markov decision processes', *Handbooks in Operations Research and Management Science*, 2(C), pp. 331–434. doi: 10.1016/S0927-0507(05)80172-0.

Puust, R.; Kapelan, Z.; Savic, D. A.; Koppel, T. (2010) 'A review of methods for leakage management in pipe networks', *Urban Water*, 7(1), pp. 25–45,.

Rajani, B. and Tesfamariam, S. (2011) 'Uncoupled axial, flexural, and circumferential pipe–soil interaction analyses of partially supported jointed water mains', *https://doi.org/10.1139/t04-048*, 41(6), pp. 997–1010. doi: 10.1139/T04-048.

Rajeev, P. ; Kodikara, Ja.; Robert, D. J.; Zeman, P.; Rajani, B. (2014) 'Factors contributing to large diameter water pipe failure'. Available at: <https://researchbank.rmit.edu.au/view/rmit:30981> (Accessed: 19 January 2023).

Rezaei, H., Ryan, B. and Stoianov, I. (2015a) 'Pipe Failure Analysis and Impact of Dynamic Hydraulic Conditions in Water Supply Networks', *Procedia Engineering*, 119(1), pp. 253–262. doi: 10.1016/J.PROENG.2015.08.883.

Rezende, D. J., Mohamed, S. and Wierstra, D. (2014) 'Stochastic Backpropagation and Approximate Inference in Deep Generative Models', in. PMLR, pp. 1278–1286. Available at: <https://proceedings.mlr.press/v32/rezende14.html> (Accessed: 9 May 2023).

Rogers, D. (2014) 'Leaking water networks: An economic and environmental disaster', in *Procedia Engineering*. doi: 10.1016/j.proeng.2014.02.157.

Roh, S. G. and Choi, H. R. (2005) 'Differential-drive in-pipe robot for moving inside urban gas pipelines', *IEEE Transactions on Robotics*, 21(1), pp. 1–17. doi: 10.1109/TRO.2004.838000.

Roman, H. T., Pellegrino, B. A. and Sigrist, W. R. (1993) 'Pipe crawling inspection robots: An

- overview', *IEEE Transactions on Energy Conversion*, 8(3), pp. 576–583. doi: 10.1109/60.257076.
- Rome, E.; Hertzberg, J.; Kirchner, F.; Licht, U.; Christaller, T. (1999) 'Towards autonomous sewer robots: the MAKRO project', *Urban Water*, 1(1), pp. 57–70. doi: 10.1016/S1462-0758(99)00012-6.
- Roslin, N. S.; Anuar, A.; Jalal, M. F. A.; Sahari, K. S. M. (2012) 'A Review: Hybrid Locomotion of In-pipe Inspection Robot', *Procedia Engineering*, 41, pp. 1456–1462. doi: 10.1016/J.PROENG.2012.07.335.
- Sadler, J. M.; Goodall, J. L.; Behl, M.; Bowes, B. D.; Morsy, M. M. (2020) 'Exploring real-time control of stormwater systems for mitigating flood risk due to sea level rise', *Journal of Hydrology*, 583, p. 124571. doi: 10.1016/J.JHYDROL.2020.124571.
- Saldarriaga, J. and Salcedo, C. A. (2015a) 'Determination of optimal location and settings of pressure reducing valves in water distribution networks for minimizing water losses', in *Procedia Engineering*. doi: 10.1016/j.proeng.2015.08.986.
- Saldarriaga, J. and Salcedo, C. A. (2015b) 'Determination of Optimal Location and Settings of Pressure Reducing Valves in Water Distribution Networks for Minimizing Water Losses', *Procedia Engineering*, 119(1), pp. 973–983. doi: 10.1016/J.PROENG.2015.08.986.
- Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; Sutskever, I. (2017) 'Evolution Strategies as a Scalable Alternative to Reinforcement Learning'.
- Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; Abbeel, P. (2015) 'High-Dimensional Continuous Control Using Generalized Advantage Estimation', *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. Available at: <https://arxiv.org/abs/1506.02438v6> (Accessed: 27 March 2023).
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. (2017) 'Proximal Policy Optimization Algorithms'. Available at: <https://arxiv.org/abs/1707.06347v2> (Accessed: 22 September 2023).
- Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; Abbeel, P. (2014) 'Trust Region Policy Optimization', 31st International Conference on Machine Learning, ICML 2014.
- Seago, C., Bhagwan, J. and Mckenzie, R. (2007) 'Benchmarking leakage from water reticulation systems in South africa', *Water SA*, 30(5), pp. 25–32,.
- Shammas, E., Wolf, A. and Choset, H. (2006) 'Three degrees-of-freedom joint for spatial hyper-

redundant robots', *Mechanism and Machine Theory*, 41(2), pp. 170–190. Available at: https://www.academia.edu/18713751/Three_degrees_of_freedom_joint_for_spatial_hyper_redundant_robots (Accessed: 25 August 2021).

Shao, L.; Wang, Y.; Guo, B.; Chen, X. (2015) 'A review over state of the art of in-pipe robot', *2015 IEEE International Conference on Mechatronics and Automation, ICMA 2015*, pp. 2180–2185. doi: 10.1109/ICMA.2015.7237824.

Sharma, A.; Burn, S.; Gardner, T.; Gregory, A. (2010) 'Role of decentralised systems in the transition of urban water systems', *Water Supply*, 10(4), pp. 577–583. doi: 10.2166/WS.2010.187.

Shinde, P. P. and Shah, S. (2018) 'A Review of Machine Learning and Deep Learning Applications', *Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018*. doi: 10.1109/ICCUBEA.2018.8697857.

De Silva, D., Mashford, J. and Burn, S. (2011) 'Computer Aided Leak Location and Sizing in Pipe Networks'. Available at: <http://www.griffith.edu.au/> (Accessed: 2 September 2021).

Da Silva, F. L., Taylor, M. E. and Costa, A. H. R. (2018) 'Autonomously reusing knowledge in multiagent reinforcement learning', in *IJCAI International Joint Conference on Artificial Intelligence*. doi: 10.24963/ijcai.2018/774.

Silva, R. A.; Buiatti, C. M.; Buiatti, C. M.; Cruz, S. L.; Pereira, J. A.F.R. (1996) 'Pressure wave behaviour and leak detection in pipelines', *Computers & Chemical Engineering*, 20(SUPPL.1), pp. S491–S496. doi: 10.1016/0098-1354(96)00091-9.

Silver, D. *et al.* (2014) 'Deterministic Policy Gradient Algorithms'.

Silver, D. (2016) 'Mastering the game of Go with deep neural networks and tree search', *Nature*, 529(7587). doi: 10.1038/nature16961.

Singh, S.; Litman, D.; Kearns, M.; Walker, M. (2002) 'Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system,' , *Journal of Artificial Intelligence*, 16, pp. 105–133.

Soft Actor-Critic — Spinning Up documentation (no date). Available at: <https://spinningup.openai.com/en/latest/algorithms/sac.html#background> (Accessed: 16 November 2023).

Sokolov, Y.; Kozma, R.; Werbos, L. D.; Werbos, P. J. (2015) 'Complete stability analysis of a

heuristic approximate dynamic programming control design'. *Journal of Automatica*, 59 pp. 9-18. doi: 10.1016/j.automatica.2015.06.001

Sophocleous, S.; Savić, D. A.; Kapelan, Z.; Giustolisi, O. (2017) 'A Two-stage Calibration for Detection of Leakage Hotspots in a Real Water Distribution Network', *Procedia Engineering*, 186, pp. 168–176. doi: 10.1016/J.PROENG.2017.03.223.

Sousa, J.; Muranho, J.; Sá Marques, A.; Gomes, R. (2014) 'WaterNetGen Helps C-Town', *Procedia Engineering*, 89, pp. 103–110. doi: 10.1016/J.PROENG.2014.11.165.

Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 2.2.1 documentation (no date). Available at: <https://stable-baselines3.readthedocs.io/en/master/> (Accessed: 6 January 2024).

Strehl, A. L.; Li, L.; Wiewiora, E.; Langford, J.; Littman, M. L. (2006) 'PAC Model-Free Reinforcement Learning', in *23rd International Conference on Machine Learning*, pp. 881–888.

Sturm, R.; Gasner, K.; Wilson, T.; Preston, S. (2014) *Real Loss Component Analysis: Tool for Economic Water Loss Control*. Available at: <https://www.waterrf.org/resource/real-loss-component-analysis-tool-economic-water-loss-control-0>. (Accessed: 2 June 2021).

Sun, Z.; Wang, P.; Vuran, M. C.; Al-Rodhaan, M. A.; Al-Dhelaan, A. M.; Akyildiz, I. F. (2011) 'MISE-PIPE: Magnetic induction-based wireless sensor networks for underground pipeline monitoring', *Ad Hoc Networks*, 9(3), pp. 218–227. doi: 10.1016/J.ADHOC.2010.10.006.

Sutton, R. S.; McAllester, D.; Singh, S.; Mansour, Y. (2000) 'Policy gradient methods for reinforcement learning with function approximation', in *Advances in Neural Information Processing Systems*.

Sutton, R. S.; Mahmood, A. R.; Precup, D.; Van Hasselt, H. (2014) 'A new $Q(\lambda)$ with interim forward view and Monte Carlo equivalence', in *31st International Conference on Machine Learning, ICML 2014*.

Sutton, R. S. and Barto, A. G. (2018) *Reinforcement Learning: An Introduction*. Second. MIT Press.

Syafiie, S.; Tadeo, F.; Martinez, E.; Alvarez, T. (2011) 'Model-free control based on reinforcement learning for a wastewater treatment problem', *Applied Soft Computing*, 11(1), pp. 73–82. doi: 10.1016/J.ASOC.2009.10.018.

Teodosiu, C.; Gilca, A.-F.; Barjoveanu, G.; Fiore, S. (2018) 'Emerging pollutants removal through

advanced drinking water treatment: A review on processes and environmental performances assessment'. doi: 10.1016/j.jclepro.2018.06.247.

Tesau, C. and Tesau, G. (1995) 'Temporal difference learning and TD-Gammon', *Communications of the ACM*, 38(3), pp. 58–68. doi: 10.1145/203330.203343.

Tessler, C.; Givony, S.; Zahavy, T. Mankowitz, D. J.; Mannor, S. (2017) 'A Deep Hierarchical Approach to Lifelong Learning in Minecraft', *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), pp. 1553–1561. doi: 10.1609/AAAI.V31I1.10744.

Thornton, J. (2003) 'Managing Leakage by Managing Pressure: A Practical Approach', *Water* 21, October 20.

Thornton, J. and Lambert, a (2005) 'Progress in practical prediction of pressure: leakage, pressure: burst frequency and pressure: consumption relationships', ... of *IWA Special Conference'Leakage*.

Thornton, J. and Lambert, A. O. (no date) 'Pressure management extends infrastructure life and reduces unnecessary energy costs'.

Tian, C. H.; Yan, J. C.; Huang, J.; Wang, Y.; Kim, D. S.; Yi, T. (2012) 'Negative pressure wave based pipeline Leak Detection: Challenges and algorithms', *Proceedings of 2012 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2012*, pp. 372–376. doi: 10.1109/SOLI.2012.6273565.

Tian, W.; Liao, Z.; Zhi, G. Wang, Z. (2022) 'Combined Sewer Overflow and Flooding Mitigation Through a Reliable Real-Time Control Based on Multi-Reinforcement Learning and Model Predictive Control', *Water Resources Research*, 58(7), p. e2021WR030703. doi: 10.1029/2021WR030703.

Tian, W.; Liao, Z.; Zhang, Z.; Wu, H.; Xin, K. (2022) 'Flooding and Overflow Mitigation Using Deep Reinforcement Learning Based on Koopman Operator of Urban Drainage Systems', *Water Resources Research*, 58(7), p. e2021WR030939. doi: 10.1029/2021WR030939.

Trust Region Policy Optimization — Spinning Up documentation (no date). Available at: <https://spinningup.openai.com/en/latest/algorithms/trpo.html> (Accessed: 26 December 2023).

Tsitsifli, S. and Kanakoudis, V. (2010) 'Presenting a New User Friendly Tool to Asses the Performance Level & Calculate the Water Balance of Water Networks', in *PRE10 International Conference*. Lefkada Island, Greece: Lefkada Island.

Tsitsiklis, J. N. and Van Roy, B. (1997) 'An analysis of temporal-difference learning with function approximation', *IEEE Transactions on Automatic Control*, 42(5). doi: 10.1109/9.580874.

Tur, J. M. M. and Garthwaite, W. (2010) 'Robotic devices for water main in-pipe inspection: A survey', *Journal of Field Robotics*, 27(4), pp. 491–508. doi: 10.1002/ROB.20347.

Tzeng, E.; Devin, C.; Hoffman, J.; Finn, C.; Peng, X.; Levine, S.; Saenko, K.; Darrell, T. (2016) 'Towards Adapting Deep Visuomotor Representations from Simulated to Real Environments', *ICLR*.

UN-Water (2012) *UN World Water Development Report*. Available at: <https://www.unwater.org/publications/un-world-water-development-report-2012> (Accessed: 26 September 2023).

Usumier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. (2017) 'Episodic exploration for deep deterministic policies for starcraft micromanagement', in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; Kavukcuoglu, K. (2017) 'FeUdal Networks for Hierarchical Reinforcement Learning', *34th International Conference on Machine Learning, ICML 2017*, 7, pp. 5409–5418. Available at: <https://arxiv.org/abs/1703.01161v2> (Accessed: 9 May 2023).

Vitens (2017) *Epynet: Object-oriented wrapper for EPANET 2.1*. Available at: <https://github.com/Vitens/epynet> (Accessed: 6 May 2022).

Wai-Lok Lai, W., Dérobert, X. and Annan, P. (2018) 'A review of Ground Penetrating Radar application in civil engineering: A 30-year journey from Locating and Testing to Imaging and Diagnosis', *NDT & E International*, 96, pp. 58–78. doi: 10.1016/J.NDTEINT.2017.04.002.

Walski, T. (2003) 'Assembling A Model', in *Advanced Water Distribution Modeling and Management*. Dayton: Civil and Environmental Engineering and Engineering Mechanics Faculty, pp. 75–132.

Walski, T.; Whitman, B.; Baron, M.; Gerloff, F. (2009) 'Pressure vs. Flow Relationship for Pipe Leaks', *Proceedings of World Environmental and Water Resources Congress 2009 - World Environmental and Water Resources Congress 2009: Great Rivers*, 342, pp. 1–10. doi: 10.1061/41036(342)10.

Wan, J.; Yu, Y.; Wu, Y.; Feng, R.; Yu, N. (2012) 'Hierarchical leak detection and localization method in natural gas pipeline monitoring sensor networks', *Sensors*, 12(1), pp. 189–214. doi:

10.3390/S120100189.

Wang, Z.; Schaul, T.; Hessel, M.; Lanctot, M. (2015) 'Dueling Network Architectures for Deep Reinforcement Learning', *33rd International Conference on Machine Learning, ICML 2016*, 4, pp. 2939–2947. Available at: <https://arxiv.org/abs/1511.06581v3> (Accessed: 1 April 2023).

Waterloss (2018) *New calculEAKator coming soon*. Available at: <http://www.waterloss.com.ba/uskoro-novi-calculleakator/#more-592>. (Accessed: 18 June 2021).

Welcome to Stable Baselines3 Contrib docs! — Stable Baselines3 - Contrib 2.2.1 documentation (no date). Available at: <https://sb3-contrib.readthedocs.io/en/master/> (Accessed: 6 January 2024).

Werbos, P. J. (1977) 'Advanced Forecasting Methods for Global Crisis Warning and Models of Intelligence', *General Systems*, 22.

Westra, S.; Brown, C.; Lall, U.; Sharma, A. (2007) 'Modeling multivariable hydrological series: Principal component analysis or independent component analysis?', *Water Resources Research*, 43(6), p. 6429. doi: 10.1029/2006WR005617.

White, D. J. (Douglas J. (1969) 'Dynamic programming', p. 181. Available at: https://books.google.com/books/about/Dynamic_Programming.html?id=SnBRAAAAMAAJ (Accessed: 7 April 2023).

Williams, R. J. (1988) 'On the use of backpropagation in associative reinforcement learning', pp. 263–270. doi: 10.1109/ICNN.1988.23856.

Williams, R. J. (1992) 'Simple statistical gradient-following algorithms for connectionist reinforcement learning', *Machine Learning 1992 8:3*, 8(3), pp. 229–256. doi: 10.1007/BF00992696.

Wols, B. A., Van Daal, K. and Van Thienen, P. (2014) 'Effects of Climate Change on Drinking Water Distribution Network Integrity: Predicting Pipe Failure Resulting from Differential Soil Settlement', *Procedia Engineering*, 70, pp. 1726–1734. doi: 10.1016/J.PROENG.2014.02.190.

Wu, R.; Liao, Z.; Zhao, L.; Kong, X. (2008) 'Wavelets application on acoustic emission signal detection in pipeline', *Canadian Conference on Electrical and Computer Engineering*, pp. 1211–1214. doi: 10.1109/CCECE.2008.4564731.

Wu, Y.; Liu, S.; Wu, X.; Liu, Y.; Guan, Y. (2016) 'Burst detection in district metering areas using a

- data driven clustering algorithm', *Water Research*, 100, pp. 28–37. doi: 10.1016/J.WATRES.2016.05.016.
- Wu, Y. and Liu, S. (2017) 'A review of data-driven approaches for burst detection in water distribution systems', *Urban Water Journal*, 14(9), pp. 972–983. doi: 10.1080/1573062X.2017.1279191.
- Xu, J.; Wang, H.; Rao, J.; Wang, J. (2021) 'Zone scheduling optimization of pumps in water distribution networks with deep reinforcement learning and knowledge-assisted learning', *Soft Computing*, 25(23), pp. 14757–14767. doi: 10.1007/S00500-021-06177-3/FIGURES/7.
- Xu, Q.; Liu, R.; Chen, Q.; Li, R. (2014) 'Review on water leakage control in distribution networks and the associated environmental benefits', *Journal of Environmental Sciences (China)*, 26(5), pp. 955–961. doi: 10.1016/S1001-0742(13)60569-0.
- Yang, D.; Zhao, L.; Lin, Z.; Qin, T.; Bian, J.; Liu, T.-Y. (2019) 'Fully Parameterized Quantile Function for Distributional Reinforcement Learning', *Advances in Neural Information Processing Systems*, 32.
- Yang, Q.; Cao, W.; Meng, W.; Si, J. (2022) 'Reinforcement-Learning-Based Tracking Control of Waste Water Treatment Process under Realistic System Conditions and Control Performance Requirements', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(8), pp. 5284–5294. doi: 10.1109/TSMC.2021.3122802.
- Ye, G. and Fenner, R. A. (2010) 'Kalman Filtering of Hydraulic Measurements for Burst Detection in Water Distribution Systems', *Journal of Pipeline Systems Engineering and Practice*, 2(1), pp. 14–22. doi: 10.1061/(ASCE)PS.1949-1204.0000070.
- Yu, J.; Zhang, L.; Chen, J.; Xiao, Y.; Hou, D.; Huang, P.; Zhang, G.; Zhang, H. (2021) 'An Integrated Bottom-Up Approach for Leak Detection in Water Distribution Networks Based on Assessing Parameters of Water Balance Model', *Water*, 13(6), p. 867.
- Zadkarami, M., Shahbazian, M. and Salahshoor, K. (2017) 'Pipeline leak diagnosis based on wavelet and statistical features using Dempster–Shafer classifier fusion technique', *Process Safety and Environmental Protection*, 105, pp. 156–163. doi: 10.1016/J.PSEP.2016.11.002.
- Zaman, D.; Tiwari, M. K.; Gupta, A. K.; Sen, D. (2020) 'A review of leakage detection strategies for pressurised pipeline in steady-state', *Engineering Failure Analysis*, 109, p. 104264. doi: 10.1016/j.engfailanal.2019.104264.
- Zhang, Z., Zhang, D. and Qiu, R. C. (2020) 'Deep reinforcement learning for power system

applications: An overview', *CSEE Journal of Power and Energy Systems*, 6(1), pp. 213–225. doi: 10.17775/CSEEJPES.2019.00920.

Zhao, W., Queralta, J. P. and Westerlund, T. (2020) 'Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey', *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 737–744. doi: 10.1109/SSCI47803.2020.9308468.

Zhou, X.; Tang, Z.; Xu, W.; Meng, F.; Chu, X.; Xin, K.; Fu, G. (2019) 'Deep learning identifies accurate burst locations in water distribution networks', *Water Research*, 166, p. 115058. doi: 10.1016/J.WATRES.2019.115058.

Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. (2016) 'Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning', *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3357–3364. doi: 10.1109/ICRA.2017.7989381.

Ziebart, B. D. and Fox, D. (2010) 'Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy'.

Appendices

Appendix A: WDN-DRL Environment Code

Below is the code used to develop the WDN-DRL Environment used to communicate between the agents and the hydraulic software.

```
#Import block
from cmath import inf
import numpy as np
import pandas as pd
import networkx as nx
import random
import os
import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnchoredText
from matplotlib.animation import PillowWriter
import scipy.stats as stats
from scipy.optimize import minimize as nm
from epynet import Network
from gym import Env
from gym.spaces import Box
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import dummy_vec_env
from stable_baselines3.common.evaluation import evaluate_policy

class WaterNetwork(Env):
    """
    Title: Water Distribution Network - Deep Reinforcement Learning Ecosystem
    Description: A class designed to allow optimisation algorithms and DRL agents to interact with
    hydraulic files in epanet. Compatible hydraulic files include .inp files and .net
    Version: 3.0
    Notes: Report function completed and all data is exported
    """
```

Action space is modified to allow all valves to be controlled at once. This version includes more render functions (states, waterloss, reward per junction, interactive map)

This version does not include the wrong move penalty or max reward. Just simply calculates the difference in penalties

This version evaluates only one timestep per episode and trains the neural network on the available 24hrs'''

```
'''
:param wdn_name: The name of the WDN hydraulic file to be imported
:param episode_len: The number of steps permissible per episode
:param freeze: If true, the next step of the episode is paused until prompted externally
:param burst: Whether it is a burst event or background leakage
:param n_junc: number of burst junctions
:param emitter_exp: The emitter exponent in the leakage rate equation
:param reset_orig_settings: Whether each episode should start with the original settings or
follow on from the previous episode
:param reset_orig_junc: Whether the burst junctions should stay the same every episode
or change randomly at the start of each episode.
:param orig_junc: The list of original burst junctions uid.
:param setmin: Value of the minimum acceptable setting in the action space
:param setmax: Value of the maximum acceptable setting in the action space
:param scale: List of reward scale ratios
:param seed: Seed for randomness
'''
```

'''Initialisation function defining necessary parameters'''

```
def __init__(self,
              wdn_name      = 'd-town',
              episode_len   = 24,
              freeze        = False,
              burst         = False,
              n_junc        = 0,
              emitter_exp   = 1.18,
              reset_orig_settings = True,
              reset_orig_junc  = False,
```

```

    orig_junc      = [],
    setmin         = 0.0,
    setmax         = 70.0,
    scale          = [1,1],
    seed           = None):

#Initiating seed
self.seedNum = seed
if self.seedNum:
    np.random.seed(self.seedNum)
else:
    np.random.seed()

#Using path and os to get the corret path to the water network
self.wdn_name = wdn_name
pathToRoot = os.path.dirname(os.path.realpath('__file__'))
pathToWdn = os.path.join(pathToRoot,'Epanet Networks', wdn_name + '.inp')
self.network = Network(pathToWdn)
self.network.ep.ENsetoption(4, emitter_exp)

#Defining nodes, pipes, valves, junctions
self.nodes = self.network.nodes
self.pipes = self.network.pipes
self.valves = self.network.valves
self.junctions = self.network.junctions

#Values for duration, timestep in seconds and episode length
self.duration = episode_len
self.timestep = 3600
self.step_count = 0
self.episode = 0
self.done = False
self.freeze = freeze
self.burst = burst

```

```

#Collecting all the set points for PRVs to make the state and setting
#Selecting the valves that are PRVs

self.setmin = setmin
self.setmax = setmax

self.reset_orig_settings = reset_orig_settings
self.prv_uid = self.get_prv_uid()
self.settings = self.get_prv_setting()
self.state = pd.DataFrame()
self.orig_settings = self.settings.loc[0,:].to_dict()
self.new_state = np.NaN
self.dimensions = len(self.prv_uid)

#Action space will be a Box action space
self.action_space = Box(low=setmin, high=setmax, shape=(len(self.prv_uid),))
#Observation space
self.observation_space = Box(low=-inf, high=inf, shape=((len(self.junctions) +
len(self.prv_uid)),) )

#Introduce leak magnitude, leakage to a random junc or user-defined junc
if burst:
    self.leak_mag = 3
else:
    self.leak_mag = 0

if reset_orig_junc:
    self.leak_junc = orig_junc
else:
    self.leak_junc = []
    for i in range(n_junc):
        self.leak_junc.append(random.choice(self.junctions.uid))

self.emitter_exp = emitter_exp
self.pressure_limit = [10, 200]

```

```

#Render logs. pressure and flow logs per step for each episode.
self.violation_logs = np.zeros(episode_len)
self.state_logs1 = np.empty((episode_len,3))
self.state_logs2 = np.empty((episode_len,3))
self.reward_log = np.empty((episode_len,))
self.reward_logs2 = np.empty((episode_len, len(self.junctions)))
self.waterloss_logs1 = np.empty((episode_len,))
self.waterloss_logs2 = np.empty((episode_len,))

#reward control
self.rewscale = scale
self.laziness_penalty = 0
self.repeat_count = 0

#Dataframes for leak df, action df
self.action_df = pd.DataFrame()

'''Getting state of avg pressure, avg waterloss, avg pressure, avg flow'''
'''Getting a list of the relevant prv uids which will be the agents'''
def get_prv_uid(self):
    prv_uid = []
    for k in self.valves:
        if k.valve_type == 'PRV' or k.valve_type == 'TCV' :
            prv_uid.append(k.uid)
    return np.array(prv_uid)

'''Getting a dict of PRV set points'''
def get_prv_setting(self):
    settings_index = pd.Index([self.step_count], name='Step')
    settings = {}
    for v in self.prv_uid:
        settings[v] = self.valves[v].get_object_value(5)
    return pd.DataFrame(settings, index=settings_index)

def get_before_data(self):

```

```

self.network.reset()

if self.burst:
    for oj in self.junctions.uid:
        self.junctions[oj].set_object_value(3, 0)
    for ol in self.leak_junc:
        self.junctions[ol].set_object_value(3, self.leak_mag)
else:
    pass

for os in self.prv_uid:
    self.valves[os].set_object_value(5, self.orig_settings[os])

self.network.solve((self.step_count%24)*self.timestep)

#Populate action df
#The action dataframe is the main collection hydraulic data including junction inflows
(before and after),
#number of pressure violations (before and after), nodal pressure before and after and
leakage (before and after)
self.action_df = pd.DataFrame({
    'emitter': self.junctions.emitter, 'flow_before': self.junctions.inflow, 'vio_before':
self.num_vio(), 'pressure_before': self.junctions.pressure})
self.get_leakage_rate(self.action_df, before=True)

#The state stores main hydraulic data from each step is recorded in the state dataframes
self.state_before = pd.DataFrame(
    {'avg_pressure': self.action_df['pressure_before'].mean(), 'water_loss':
self.action_df['leak_before'].sum(), 'avg_flow': self.action_df['flow_before'].mean()},
index=[self.step_count])

"""Function used to calculate leakage rate using junction pressure measurements, emitter
coefficients and the emitter exponents"""
def get_leakage_rate(self,df, before= True):

```

```

if before:
    df['leak_before'] = df['emitter']*df['pressure_before']**self.emitter_exp

else:
    df['leak_after'] = df['emitter']*df['pressure_after']**self.emitter_exp

pass

'''Getting the flows and violations before introducing the leak'''
def solve_episode(self, df, leak_junc, leak_mag, step_count, timestep, settings):

    self.network.reset()

    #Get the values after valve settings and leak.
    #The state will be orig_state if its just a leak and state if it is action and leak
    if self.burst:
        for j in self.junctions.uid:
            self.junctions[j].set_object_value(3, 0)
        for l in leak_junc:
            self.junctions[l].set_object_value(3, leak_mag) # introducing leak magnitude to a
random leak_junc
    else:
        pass
    for s in self.prv_uid:
        self.valves[s].set_object_value(5, settings[s])

    self.network.solve((step_count%24)*timestep)
    self.get_after_data(df)

    return df

'''Getting flows and vios and waterloss after introducing a leak'''
def get_after_data(self, df):
    df['flow_after'] = self.junctions.inflow

```

```

df['vio_after'] = self.num_vio()
df['pressure_after'] = self.junctions.pressure
self.get_leakage_rate(df, False)
df['water_saved'] = df['leak_before'] - df['leak_after']

#Calculating the reward based on the violations and leakage rate difference
# Hyperbolic tan of % water saved is percent_loss will be multiplied by its reward scale
# Pressure violations is whether we have resolved any violations (+ve) or didn't resolve (0)
or created new violations (-ve)

df['percent_saved'] = np.tanh( (df['water_saved'] / df['leak_before'])*self.rewscale[0] )
#remove tanh function for Jowitt&Xu network
df['percent_vio'] = (df['vio_before'] - df['vio_after'])*self.rewscale[1]
df['reward'] = df[['percent_saved', 'percent_vio']].sum(axis=1)
df.fillna(0)
df.dropna()

#Get state (avg pressure, waterloss, avg flow)
self.state = pd.DataFrame(
    {'avg_pressure': df['pressure_after'].mean(), 'water_loss':
df['leak_after'].sum(), 'avg_flow': df['flow_after'].mean()}, index=[self.step_count])

'''Calculating if the junction is violating the pressure limits'''
def num_vio(self):
    vio2 = (self.junctions.pressure > self.pressure_limit[1])*1
    vio1 = (self.junctions.pressure < self.pressure_limit[0])*1
    vio = vio1 + vio2
    return vio

'''Step function simulates one step in each hour. First the leakless actionless data is retrieved
then the leak data is run
    and then an action is introduced to be compared to the leak data. The action and leak df
are compared to return their
    respective penalties.
    The reward is calculated as the amount of penalty saved by performing the action.
    The episode ends if the no of steps reaches 24 or if the state isn't changing.'''

```



```

def step(self, action):

    #Get your 'before' data for leak and action then get the after leak data
    self.get_before_data()

    self.state_logs1[self.step_count,:] = self.state_before.values

    #Using the action to find the corresponding move to the correct valve
    new_settings_index = pd.Index([self.step_count+1], name='Step')
    self.new_settings = pd.DataFrame(self.settings.loc[self.step_count,:].to_dict(), index =
new_settings_index)
    for i in range(len(action)):
        valve_index = self.prv_uid[i] # returns 0 for the first prv; 1 for the second and so on
        move = action[i] #when running it in a testing loop
        #Change the settings after the action. Add the leak and valve setting
        self.new_settings[valve_index] = move

    #Get the after data for the action_df. If there is no burst the leak_mag will be zero and not
influence the simulation
    self.solve_episode(
        self.action_df, self.leak_junc, self.leak_mag, self.step_count, self.timestep,
self.new_settings)

    #append setting df with the new setting
    self.settings = pd.concat([self.settings, self.new_settings])

    #Calculate rewards:
    reward = self.action_df['reward'].sum()
    if self.step_count>0:
        if self.settings.loc[self.step_count].equals(self.settings.loc[self.step_count-1]):
            self.repeat_count += 1
        else:
            self.repeat_count = 0
    if self.repeat_count >= 3:
        reward += self.laziness_penalty

```

```

#Adding metrics to logs
self.violation_logs[self.step_count] = self.action_df['vio_after'].sum()
self.waterloss_logs1[self.step_count] = self.action_df['leak_before'].sum()
self.waterloss_logs2[self.step_count] = self.action_df['leak_after'].sum()
self.reward_log[self.step_count] = reward
self.reward_logs2[self.step_count,:] = self.action_df['reward'].to_numpy()
self.state_logs1[self.step_count,:] = self.state_before.values
self.state_logs2[self.step_count,:] = self.state.values

self.observation = self.get_observation()

#Increase the step count, episode, and check if we reached duration then we reset.
#There are 24 steps in each episode signifying the 24 hours of the day
if not self.freeze:
    self.step_count += 1
if self.step_count == self.duration or self.done:
    self.done = True
    self.episode += 1
    self.step_count = 0
    self.setting_logs = self.settings
    if self.reset_orig_settings:
        self.settings = pd.DataFrame(self.orig_settings, index=pd.Index([self.step_count],
name='Step'))

    return self.observation, reward, self.done, {}

"""Function to send the reward to SciPy optimisation algorithms"""
def reward_to_scipy(self, action):

    obs, reward, done, info = self.step(action)
    return -reward

"""Function to send the reward to SciPy optimisation algorithms"""
def reward_to_deap(self, action):

    obs, reward, done, info = self.step(action)

```

```

return reward

"""Calculate the observation for the current step"""
def get_observation(self):

    obs = self.action_df['pressure_before']
    obs = pd.concat([obs, self.settings.loc[self.step_count]], axis=0)

    return obs.transpose()

"""Environment resets. Need to calculate an appropriate first observation"""
def reset(self):

    self.done = False
    self.step_count = 0
    self.prevReward = 0
    self.get_before_data()
    self.observation = self.get_observation()

    return self.observation

"""Data visualisation functions"""
def render(self, choice):
    #Choice is a menu of possible render functions
    menu = {'interactive map': 0, 'reward across junctions': 1, 'settings': 2, 'water loss': 3,
'states': 4}

    #Interactive map
    if menu[choice] == 0:

        fig = plt.figure()
        metadata = dict(title=f'{self.wdn_name} interactive map - ep{self.episode}',
artist='Ahmed Negm')
        writer = PillowWriter(fps=2, metadata=metadata) #pillow writer to make gif

```

```

with writer.saving(fig, f'{self.wdn_name} interactive map - ep{self.episode}.gif', 150):
#figure to capture, title and fps
    for s in range(self.duration):
        valve_list = []
        downstream_uid = []
        upstream_uid = []

        for p in self.prv_uid:
            valve_list.append((self.valves[p].downstream_node.uid,
self.valves[p].upstream_node.uid))
            for i in self.pipes.uid:
                downstream_uid.append(self.pipes[i].downstream_node.uid)
            for i in self.pipes.uid:
                upstream_uid.append(self.pipes[i].upstream_node.uid)

        G = nx.Graph()
        for h in range(len(self.nodes)):
            G.add_node(self.nodes.uid[h])
        for i in range(len(self.pipes)):
            G.add_edge(downstream_uid[i], upstream_uid[i])
        #for p in self.prv_uid:
            # G.add_edge(self.valves[p].downstream_node.uid,
self.valves[p].upstream_node.uid, weight=self.setting_logs.loc[s,p].round(1))

        edge_labels = nx.get_edge_attributes(G,'weight')

        leak_options = {'node_color' : 'red', 'node_size' : 10,}
        valve_options = {'edge_color' : 'green'}
        options = {'node_color': 'blue', 'node_size': 10}

        nx.draw(G, self.nodes.coordinates, with_labels=False, **options)
        nx.draw_networkx_nodes(G, self.nodes.coordinates, nodelist= self.leak_junc,
**leak_options)
        nx.draw_networkx_edges(G, self.nodes.coordinates, edgelist=valve_list,
**valve_options )

```

```

    nx.draw_networkx_edge_labels(G, self.nodes.coordinates, edge_labels)
    plt.title('Interactive map')

    writer.grab_frame()
    plt.clf()

    print('Render Completed')
    pass

    #reward across junctions
    elif menu[choice] == 1:
        #reward across junctions gif. This render function creates a gif of the penalties available
        in the network across the span of a 24h episode

        #The leak nodes are labelled with a 'o'. The total reward saved and step count is
        displayed in the top-right corner.

        #The leak penalties are red and logged at self.reward_logs2. The action penalties are
        blue and logged at self.reward_logs1.

        fig = plt.figure()
        metadata = dict(title='Junc reward spread over day', artist='Ahmed Negm')
        writer = PillowWriter(fps=2, metadata=metadata) #pillow writer to make gif
        with writer.saving(fig, f'Junc reward render ep{self.episode}.gif', 150): #figure to
        capture, title and fps

            for i in range(self.duration): #for each step in the duration of the episode (24h)
                ax = plt.gca()

                #Plot action penalties against the index

                plt.bar(self.junctions.uid, self.reward_logs2[i,:], alpha=0.5, label='Reward after',
                color='blue')

                #Plot legend, title, labels, and leakage markers

                plt.legend(loc='upper left')

                plt.title('Reward across junctions')

                for l in self.leak_junc:
                    plt.plot(l, 0, 'o', ms=5, mec='k', mfc='none', mew=0.5)

                plt.xlabel('Junctions')
                plt.ylabel('Reward')

```

```

        #Plot anchored text describing the reward (reward saved) and step count on the
upper right corner
        at = AnchoredText(f'Reward: {self.reward_log[i].round(2)} \nStep: {i+1}',
                            prop=dict(size=10), frameon=True, loc='upper right')
        at.patch.set_boxstyle('round,pad=0.,rounding_size=0.2')
        ax.add_artist(at)

        #Capture the final figure for the animation then clear
        writer.grab_frame()
        plt.clf()

    print('Render Complete')
    pass

#Settings
elif menu[choice] == 2:

    plt.figure()
    x_data = range(self.duration+1)
    for i in self.prv_uid: #for each step in the duration of the episode (24h)
        plt.plot(x_data, self.setting_logs[i], label=i)
    plt.legend()
    plt.title('Settings changing across episode')
    plt.xlabel('Step Count')
    plt.ylabel('Setting')
    plt.savefig(f'Settings across ep{self.episode}')
    plt.show()

#Water Loss
elif menu[choice] == 3:

    x_data = range(self.duration)
    plt.figure()
    plt.plot(x_data, self.waterloss_logs1, label='Leakage before', color='red')
    plt.plot(x_data, self.waterloss_logs2, label='Leakage after', color='blue')
    plt.legend()

```

```

plt.title('Waterloss across steps')
plt.xlabel('Step Count')
plt.ylabel('Water Loss in Litres')
plt.savefig(f'Water Loss of ep{self.episode}')
plt.show()

print('Render Complete')

#States
elif menu[choice] == 4:

    fig, ax = plt.subplots(3)
    x_data = range(self.duration)
    labels = ['Average Pressure', 'Water Loss', 'Average Flow']

    for i in range(3):
        ax[i].plot(x_data, self.state_logs2[:,i], label=labels[i], color='blue')
        ax[i].plot(x_data, self.state_logs1[:,i], label=labels[i], color='red')
        ax[i].set(xlabel='Step Count', ylabel=labels[i])

    fig.suptitle('States across steps')
    plt.savefig(f'States of ep{self.episode}')
    plt.show()

pass

'''Function for creating reports based on algorithm performance'''
def report(self, choice):
    menu = {'excel': 0, 'html': 1, 'pdf': 2}
    path = os.path.join(os.path.dirname(os.path.realpath('__file__')), 'Excel Reports', f'WN2.5
{self.wdn_name} Report - Ep {self.episode} - Step {self.step_count}.xlsx')
    ep_path = os.path.join(os.path.dirname(os.path.realpath('__file__')), 'Excel Reports',
f'WN2.5 {self.wdn_name} Report - Ep {self.episode} Logs.xlsx')

#Excel

```

```

if menu[choice] == 0:

    """
    Logging all the data into excel files using Excel writer
    Each step's data will be exported in one file including the action df, settings,
observations and states
    """

    with pd.ExcelWriter(path) as writer:

        self.action_df.to_excel(writer, sheet_name=f'Master df Step {self.step_count}',
startcol=0, startrow=1)

        self.settings.to_excel(writer, sheet_name=f'Master df Step {self.step_count}',
startcol=15, startrow=1)

        self.observation.to_excel(writer, sheet_name=f'Obs and state Step {self.step_count}',
startcol=0, startrow=1)

        self.state.to_excel(writer, sheet_name=f'Obs and state Step {self.step_count}',
startcol=5, startrow=1)

        worksheet = writer.sheets[f'Master df Step {self.step_count}']
        worksheet.write_string(0,0,'Action DataFrame')
        worksheet.write_string(0,15,'Settings')

        worksheet = writer.sheets[f'Obs and state Step {self.step_count}']
        worksheet.write_string(0,0,'Observation data')
        worksheet.write_string(0,5,'State data')

    #Episode summary report
    if self.step_count == 23:
        with pd.ExcelWriter(ep_path) as writer:
            state_log1_df = pd.DataFrame(self.state_logs1, columns=['Average Pressure',
'Water Loss', 'Average Flow'])

            state_log2_df = pd.DataFrame(self.state_logs2, columns=['Average Pressure',
'Water Loss', 'Average Flow'])

            reward_logs2_df = pd.DataFrame(self.reward_logs2, columns=self.junctions.uid)

```



```

reward_sums_df = pd.DataFrame({
    'Water Saved': self.state_logs1[:,1]-self.state_logs2[:,1],
    'Water Saved %': (self.state_logs1[:,1]-
self.state_logs2[:,1])*100/self.state_logs1[:,1],
    'Rewards': self.reward_log,
    'Violations': self.violation_logs
})

state_log1_df.to_excel(writer, sheet_name=f'Episode {self.episode}', startrow=1)
state_log2_df.to_excel(writer, sheet_name=f'Episode {self.episode}', startcol=0,
startrow=28)
reward_logs2_df.to_excel(writer, sheet_name=f'Episode {self.episode}', startcol=5,
startrow=1)
reward_sums_df.to_excel(writer, sheet_name=f'Episode {self.episode}', startcol=5,
startrow=28)

worksheet = writer.sheets[f'Episode {self.episode}']
worksheet.write_string(0,0,'States before action')
worksheet.write_string(27,0,'States after action')
worksheet.write_string(0,5,'Junction rewards')

    pass
elif menu[choice] == 1:
    pass
elif menu[choice] == 2:
    pass

```

Appendix B: Optimisation Algorithms Code

In this appendix, we outline the python scripts used to create the benchmark optimisation algorithms (Nelder Mead, Particle Swarm Optimisation, and Differential Evolution). The libraries required for these algorithms are imported below.

```

#Import block
from cmath import inf
from typing import Callable

```

```

import math
import numpy as np
import pandas as pd
import random
import operator
import os
from deap import base
from deap import creator
from deap import tools
from scipy.optimize import minimize as nm
import tensorflow as tf
from WaterNetwork import WaterNetwork

```

In these algorithms, the 'env' parameter is a placeholder for the WDN-DRL environment file (Appendix A).

```

env = WaterNetwork(wdn_name='SZ08_PEAK', burst=True,
reset_orig_junc=True, orig_junc=test_leak_junc, n_junc=32, scale=[5,1])
env_run = WaterNetwork(wdn_name='SZ08_PEAK', burst=True,
reset_orig_junc=False, n_junc=32, scale=[5,1], freeze=True)
df_header = ['index', 'reward', 'evals']

```

This is used to create the Nelder Mead class used in the case studies below.

```

class nelder_mead_method():

    def __init__(self):
        self.options = { 'maxfev': 1000, 'xatol' : .005, 'fatol' : .01,
'disp': True}

    def maximise(self, id):

        init_guess = []
        for i in range(len(env_run.prv_uid)):
            init_guess.append(random.randint(0,70))

        bounds = []
        for i in range(len(env_run.prv_uid)):

```

```

        bounds.append((0,70))

    env_run.step_count = id
    result = nm(env_run.reward_to_scipy, init_guess,
options=self.options, bounds=bounds, method='Nelder-Mead')
    result_df = pd.DataFrame(np.empty((1,len(df_header))),
columns=df_header)
    env_run.settings.drop(index=env_run.step_count+1)

    result_df[df_header[0]] = id
    result_df[df_header[1]] = -result.fun
    result_df[df_header[2]] = result.nit
    print(f'x={result.x} y={result.fun}')
    action = result.x

    for i in range(len(env.prv_uid)):
        result_df['Setting of '+str(env.prv_uid[i])] = result.x[i]

    return result_df, action

```

The particles swarm optimisation class is included in the same Jupyter notebook below.

```

class pso():

    def __init__(self):

        #Creator setup
        creator.create("FitnessMax", base.Fitness, weights=(1.0,))
        creator.create("Particle", list, fitness=creator.FitnessMax,
speed=list, smin=None, smax=None, best=None)
        pass

        #Generating particles function. Parameters include minimum and max
speed of particle (smin, smax); minimum and maximum positions (pmin,
pmax) and size
        #Essentially smax - smin makes themaximum step size
        def generate_part(self, size, smin, smax, pmin, pmax):

```

```

    part = creator.Particle(random.uniform(pmin, pmax) for _ in
range(size))
    part.speed = [random.uniform(smin, smax) for _ in range(size)]
    part.smin = smin
    part.smax = smax
    return part

#Update the particle's after each search
#part is the parameter for the particles being updated
#Phi1 is the parameter for the particle's probability to exploit it's
best score this maps into a random distribution for each particle in list
u1
#Phi2 is the parameter for the particle's probability to explore the
global best score this maps into a random distribution for each particle
in lise u
#v_u1 is the exploitation velocity which is u1 x the difference
between best position and the part's current position
#v_u2 is the exploratory velocity which is u2 x the differenc
between the global best position and the part's current position
def update_part(self, part, best, phi1, phi2,setmin, setmax):
    u1 = (random.uniform(0, phi1) for _ in range(len(part)))
    u2 = (random.uniform(0, phi2) for _ in range(len(part)))
    v_u1 = map(operator.mul, u1, map(operator.sub, part.best, part))
    v_u2 = map(operator.mul, u2, map(operator.sub, best, part))

#Each particle's speed is calculated by adding its current speed
to v_u1 and v_u2
#Then the speeds are clipped to keep the absolute speed between
smin and smax
#copysign copies the sign of the second parameter to the first
parameter
part.speed = list(map(operator.add, part.speed, map(operator.add,
v_u1, v_u2)))
for i, speed in enumerate(part.speed):
    if abs(speed) < part.smin:
        part.speed[i] = math.copysign(part.smin, speed)
    elif abs(speed) > part.smax:
        part.speed[i] = math.copysign(part.smax, speed)

```

```

        #Each particle's new position is calculated by adding the current
position to the speed

        #The positions are clipped to ensure that the particle position
is within our valve setting limits setmin and setmax

        tmp2 = np.asarray(list(map(operator.add, part, part.speed)))
        np.clip(a=tmp2, a_min=setmin, a_max=setmax, out=tmp2)
        part[:] = tmp2.tolist()

def maximise(self, id):
    mu = 30 #number of particles
    ngen = 20 #number of generations/iterations
    best = None #Initialise the best particle

    #Toolbox
    toolbox = base.Toolbox()
    toolbox.register("particle", self.generate_part,
size=env_run.dimensions, pmin=env_run.setmin, pmax=env_run.setmax, smin=-
1, smax=1)
    toolbox.register("population", tools.initRepeat, list,
toolbox.particle)
    toolbox.register("update", self.update_part, phi1=0.5, phi2=0.5,
setmin=env_run.setmin, setmax=env_run.setmax)
    toolbox.register("evaluate", env_run.reward_to_deep)

    #Initiate population and statistics
    pop = toolbox.population(n=mu)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", np.mean)
    stats.register("std", np.std)
    stats.register("min", np.min)
    stats.register("max", np.max)

    #Running optimisation block
    env_run.step_count = id
    for g in range(ngen): #for each iteration and each particle in
the population
        for part in pop:

```

```

env_run.settings.drop(index=env_run.step_count)
part.fitness.values = [toolbox.evaluate(part).tolist()]
#Evaluates the fitness of the part by reading the reward
    if not part.best or part.best.fitness < part.fitness: #If
this is the first move or the best position yet we update the best
position for this particle
        part.best = creator.Particle(part)
        part.best.fitness.values = part.fitness.values
    if not best or best.fitness < part.fitness: #If this the
first move in the entire population or the best global position we update
the best global position
        best = creator.Particle(part)
        best.fitness.values = part.fitness.values
    for part in pop:
        toolbox.update(part, best) #update the particle and best
particle for the next iteration

    result_df = pd.DataFrame(np.empty((1, len(df_header))),
columns=df_header)
    result_df[df_header[0]] = id
    result_df[df_header[1]] = best.fitness.values[0]
    result_df[df_header[2]] = ngen*mu
    print(f'x={best} y={best.fitness.values[0]}')

    for i in range(len(env.prv_uid)):
        result_df['Setting of '+str(env.prv_uid[i])] = best[i]

    return result_df, best

```

Finally, the differential evolution class uses the DEAP library to create and mutate the generations as shown below.

```

class de():
    def __init__(self):
        pass

    def maximise(self, id):

```

```

cr = 0.25
f = 1
mu = 30
ngen = 20

creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", np.ndarray,
fitness=creator.FitnessMax)

toolbox = base.Toolbox()

toolbox.register("attr_float", random.uniform, env_run.setmin,
env_run.setmax)
toolbox.register("individual", tools.initRepeat,
creator.Individual, toolbox.attr_float, env_run.dimensions)
toolbox.register("population", tools.initRepeat, list,
toolbox.individual)
toolbox.register("select", tools.selRandom, k=3)
toolbox.register("evaluate", env_run.reward_to_deep)

#Initiate population and statistics
pop = toolbox.population(n=mu) #A population of 30 individual
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean)
stats.register("std", np.std)
stats.register("min", np.min)
stats.register("max", np.max)

#Running optimisation block
env_run.step_count = id
hof = tools.HallOfFame(1, similar=np.array_equal) #1 individual
in the hall of fame (the best)

for g in range(1, ngen): #For the number of generations (20)
    for k, agent in enumerate(pop): #For each of the 30
individuals
        #env_run.settings.drop(index=env_run.step_count+1)

```

```

        a,b,c = toolbox.select(pop) #Randomly select three
individuals from the population
        y = toolbox.clone(agent)
        index = random.randrange(env_run.dimensions)
        for i, value in enumerate(agent):
            if i == index or random.random() < cr:
                candidate = a[i] + f*(b[i]-c[i])
                if candidate < env_run.setmin:
                    y[i] = env_run.setmin
                elif candidate > env_run.setmax:
                    y[i] = env_run.setmax
                else:
                    y[i] = candidate
            y.fitness.values = [toolbox.evaluate(y).tolist()]
            if y.fitness > agent.fitness:
                pop[k] = y
        hof.update(pop)

    result_df = pd.DataFrame(np.empty((1,len(df_header))),
columns=df_header)
    result_df[df_header[0]] = id
    result_df[df_header[1]] = hof[0].fitness.values[0]
    result_df[df_header[2]] = ngen*mu
    print(f'x={hof} y={hof[0].fitness.values[0]}')

    for i in range(len(env.prv_uid)):
        result_df['Setting of '+str(env.prv_uid[i])] = hof[0][i]
    del creator.FitnessMax, creator.Individual

    return result_df, hof[0]

```


Appendix C: DRL Algorithm Training Scripts

In this appendix, we demonstrate the training blocks used in the Jupyter python notebooks to develop the policies of the DRL algorithms.

First, we import the libraries using the code below.

```
#Import block
from cmath import inf
from typing import Callable
import math
import numpy as np
import pandas as pd
import random
import operator
import os
from deap import base
from deap import creator
from deap import tools
from scipy.optimize import minimize as nm
from stable_baselines3 import PPO, A2C, DDPG, TD3, SAC
from sb3_contrib import RecurrentPPO, TRPO, ARS, TQC
from stable_baselines3.common.results_plotter import load_results, ts2xy
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.callbacks import BaseCallback

import tensorflow as tf
from WaterNetwork import WaterNetwork
```

We rely on the SB3 library to access the agents however we also develop some callbacks to evaluate the algorithm's training process. The code for the callbacks is mentioned below.

```
class SaveOnBestTrainingRewardCallback(BaseCallback):
    """
    Callback for saving a model (the check is done every ``check_freq``
    steps)
    based on the training reward (in practice, we recommend using
    ``EvalCallback``).
```

```

:param check_freq:
:param log_dir: Path to the folder where the model will be saved.
    It must contains the file created by the ``Monitor`` wrapper.
:param verbose: Verbosity level: 0 for no output, 1 for info
messages, 2 for debug messages
"""
def __init__(self, check_freq: int, log_dir: str, verbose: int = 1):
    super(SaveOnBestTrainingRewardCallback, self).__init__(verbose)
    self.check_freq = check_freq
    self.log_dir = log_dir
    self.save_path = os.path.join(log_dir, "best_model")
    self.best_mean_reward = -np.inf

def _init_callback(self) -> None:
    # Create folder if needed
    if self.save_path is not None:
        os.makedirs(self.save_path, exist_ok=True)

def _on_step(self) -> bool:
    if self.n_calls % self.check_freq == 0:

        # Retrieve training reward
        x, y = ts2xy(load_results(self.log_dir), "timesteps")
        if len(x) > 0:
            # Mean training reward over the last 100 episodes
            mean_reward = np.mean(y[-100:])
            if self.verbose >= 1:
                print(f"Num timesteps: {self.num_timesteps}")
                print(f"Best mean reward: {self.best_mean_reward:.2f} -
Last mean reward per episode: {mean_reward:.2f}")

            # New best model, you could save the agent here
            if mean_reward > self.best_mean_reward:
                self.best_mean_reward = mean_reward
                # Example for saving best model
                if self.verbose >= 1:
                    print(f"Saving new best model to {self.save_path}")
                self.model.save(self.save_path)

```

```
return True
```

During hyperparameter training, some of the algorithms showed a better result by implementing a linear schedule to the learning rate parameter. This allows the algorithm to slowly move from exploration-based behaviour to exploitation-based behaviour as the optimal policy is developed. The code used for that is shown below.

```
def linear_schedule(initial_value: float) -> Callable[[float], float]:
    """
    Linear learning rate schedule.

    :param initial_value: Initial learning rate.
    :return: schedule that computes
        current learning rate depending on remaining progress
    """
    def func(progress_remaining: float) -> float:
        """
        Progress will decrease from 1 (beginning) to 0.

        :param progress_remaining:
        :return: current learning rate
        """
        return progress_remaining * initial_value

    return func
```

Each DRL algorithm's default hyperparameters and brief explanation is listed below as per the SB3 documentation (*Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 2.2.1 documentation, no date; Welcome to Stable Baselines3 Contrib docs! — Stable Baselines3 - Contrib 2.2.1 documentation, no date*). Following this, we show the training and evaluation of each algorithm using the following.

TRPO

```
class sb3_contrib.trpo.TRPO(policy, env, Learning_rate=0.001,
n_steps=2048, batch_size=128, gamma=0.99, cg_max_steps=15,
cg_damping=0.1, line_search_shrinking_factor=0.8,
line_search_max_iter=10, n_critic_updates=10, gae_Lambda=0.95,
```

```

use_sde=False, sde_sample_freq=-1, rollout_buffer_class=None,
rollout_buffer_kwargs=None, normalize_advantage=True, target_kl=0.01,
sub_sampling_factor=1, stats_window_size=100, tensorboard_log=None,
policy_kwargs=None, verbose=0, seed=None, device='auto',
_init_setup_model=True)

```

Where the hyperparameters are described as follows.

- **policy** (*ActorCriticPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable[[float], float]*) – The learning rate for the value function, it can be a function of the current progress remaining (from 1 to 0)
- **n_steps** (*int*) – The number of steps to run for each environment per update (i.e. rollout buffer size is n_steps * n_envs where n_envs is number of environment copies running in parallel) NOTE: n_steps * n_envs must be greater than 1 (because of the advantage normalization) See <https://github.com/pytorch/pytorch/issues/29372>
- **batch_size** (*int*) – Minibatch size for the value function
- **gamma** (*float*) – Discount factor
- **cg_max_steps** (*int*) – maximum number of steps in the Conjugate Gradient algorithm for computing the Hessian vector product
- **cg_damping** (*float*) – damping in the Hessian vector product computation
- **line_search_shrinking_factor** (*float*) – step-size reduction factor for the line-search (i.e., $\theta_{\text{new}} = \theta + \alpha^i * \text{step}$)
- **line_search_max_iter** (*int*) – maximum number of iteration for the backtracking line-search
- **n_critic_updates** (*int*) – number of critic updates per policy update
- **gae_lambda** (*float*) – Factor for trade-off of bias vs variance for Generalized Advantage Estimator
- **use_sde** (*bool*) – Whether to use generalized State Dependent Exploration (gSDE) instead of action noise exploration (default: False)
- **sde_sample_freq** (*int*) – Sample a new noise matrix every n steps when using gSDE Default: -1 (only sample at the beginning of the rollout)
- **rollout_buffer_class** (*Type[RolloutBuffer] | None*) – Rollout buffer class to use. If None, it will be automatically selected.

- **rollout_buffer_kwargs** (*Dict[str, Any] | None*) – Keyword arguments to pass to the rollout buffer on creation.
- **normalize_advantage** (*bool*) – Whether to normalize or not the advantage
- **target_kl** (*float*) – Target Kullback-Leibler divergence between updates. Should be small for stability. Values like 0.01, 0.05.
- **sub_sampling_factor** (*int*) – Sub-sample the batch to make computation faster see p40-42 of John Schulman thesis <http://joschu.net/docs/thesis.pdf>
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over.
- **tensorboard_log** (*str | None*) – the log location for tensorboard (if None, no logging)
- **policy_kwargs** (*Dict[str, Any] | None*) – additional arguments to be passed to the policy on creation.
- **verbose** (*int*) – the verbosity level: 0 no output, 1 info, 2 debug.
- **seed** (*int | None*) – Seed for the pseudo random generators
- **device** (*device | str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance.

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = TRPO('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
TRPO_path = os.path.join('Training', 'Saved Models', 'TRPO_env')
model.save(TRPO_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

PPO

```
class stable_baselines3.ppo.PPO(policy, env, learning_rate=0.0003,
n_steps=2048, batch_size=64, n_epochs=10, gamma=0.99, gae_lambda=0.95,
clip_range=0.2, clip_range_vf=None, normalize_advantage=True,
```

```
ent_coef=0.0, vf_coef=0.5, max_grad_norm=0.5, use_sde=False,
sde_sample_freq=-1, rollout_buffer_class=None,
rollout_buffer_kwargs=None, target_kl=None, stats_window_size=100,
tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None,
device='auto', _init_setup_model=True)
```

Where the hyperparameters are described as follows.

- **policy** (*ActorCriticPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable[[float], float]*) – The learning rate, it can be a function of the current progress remaining (from 1 to 0)
- **n_steps** (*int*) – The number of steps to run for each environment per update (i.e. rollout buffer size is n_steps * n_envs where n_envs is number of environment copies running in parallel) NOTE: n_steps * n_envs must be greater than 1 (because of the advantage normalization) See <https://github.com/pytorch/pytorch/issues/29372>
- **batch_size** (*int*) – Minibatch size
- **n_epochs** (*int*) – Number of epoch when optimizing the surrogate loss
- **gamma** (*float*) – Discount factor
- **gae_lambda** (*float*) – Factor for trade-off of bias vs variance for Generalized Advantage Estimator
- **clip_range** (*float | Callable[[float], float]*) – Clipping parameter, it can be a function of the current progress remaining (from 1 to 0).
- **clip_range_vf** (*None | float | Callable[[float], float]*) – Clipping parameter for the value function, it can be a function of the current progress remaining (from 1 to 0). This is a parameter specific to the OpenAI implementation. If None is passed (default), no clipping will be done on the value function. IMPORTANT: this clipping depends on the reward scaling.
- **normalize_advantage** (*bool*) – Whether to normalize or not the advantage
- **ent_coef** (*float*) – Entropy coefficient for the loss calculation
- **vf_coef** (*float*) – Value function coefficient for the loss calculation
- **max_grad_norm** (*float*) – The maximum value for the gradient clipping
- **use_sde** (*bool*) – Whether to use generalized State Dependent Exploration (gSDE) instead of action noise exploration (default: False)

- **sde_sample_freq** (*int*) – Sample a new noise matrix every *n* steps when using gSDE
Default: -1 (only sample at the beginning of the rollout)
- **rollout_buffer_class** (*Type[RolloutBuffer] | None*) – Rollout buffer class to use. If None, it will be automatically selected.
- **rollout_buffer_kwargs** (*Dict[str, Any] | None*) – Keyword arguments to pass to the rollout buffer on creation
- **target_kl** (*float | None*) – Limit the KL divergence between updates, because the clipping is not enough to prevent large update see issue #213 (cf <https://github.com/hill-a/stable-baselines/issues/213>) By default, there is no limit on the kl div.
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over.
- **tensorboard_log** (*str | None*) – the log location for tensorboard (if None, no logging)
- **policy_kwargs** (*Dict[str, Any] | None*) – additional arguments to be passed to the policy on creation.
- **verbose** (*int*) – Verbosity level: 0 for no output, 1 for info messages (such as device or wrappers used), 2 for debug messages.
- **seed** (*int | None*) – Seed for the pseudo random generators
- **device** (*device | str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance.

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = PPO('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
PPO_path = os.path.join('Training', 'Saved Models', 'PPO_env')
model.save(PPO_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

Recurrent PPO

```
class sb3_contrib.ppo_recurrent.RecurrentPPO(policy, env,
      Learning_rate=0.0003, n_steps=128, batch_size=128, n_epochs=10,
      gamma=0.99, gae_lambda=0.95, clip_range=0.2, clip_range_vf=None,
      normalize_advantage=True, ent_coef=0.0, vf_coef=0.5, max_grad_norm=0.5,
      use_sde=False, sde_sample_freq=-1, target_kl=None, stats_window_size=100,
      tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None,
      device='auto', _init_setup_model=True)
```

Where the hyperparameters are described as follows.

- **policy** (*ActorCriticPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable[[float], float]*) – The learning rate, it can be a function of the current progress remaining (from 1 to 0)
- **n_steps** (*int*) – The number of steps to run for each environment per update (i.e. batch size is $n_steps * n_env$ where n_env is number of environment copies running in parallel)
- **batch_size** (*int | None*) – Minibatch size
- **n_epochs** (*int*) – Number of epoch when optimizing the surrogate loss
- **gamma** (*float*) – Discount factor
- **gae_lambda** (*float*) – Factor for trade-off of bias vs variance for Generalized Advantage Estimator
- **clip_range** (*float | Callable[[float], float]*) – Clipping parameter, it can be a function of the current progress remaining (from 1 to 0).
- **clip_range_vf** (*None | float | Callable[[float], float]*) – Clipping parameter for the value function, it can be a function of the current progress remaining (from 1 to 0). This is a parameter specific to the OpenAI implementation. If None is passed (default), no clipping will be done on the value function. IMPORTANT: this clipping depends on the reward scaling.
- **normalize_advantage** (*bool*) – Whether to normalize or not the advantage
- **ent_coef** (*float*) – Entropy coefficient for the loss calculation
- **vf_coef** (*float*) – Value function coefficient for the loss calculation
- **max_grad_norm** (*float*) – The maximum value for the gradient clipping

- **target_kl** (*float / None*) – Limit the KL divergence between updates, because the clipping is not enough to prevent large update see issue #213 (cf <https://github.com/hill-a/stable-baselines/issues/213>) By default, there is no limit on the kl div.
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over
- **tensorboard_log** (*str / None*) – the log location for tensorboard (if None, no logging)
- **policy_kwargs** (*Dict[str, Any] / None*) – additional arguments to be passed to the policy on creation
- **verbose** (*int*) – the verbosity level: 0 no output, 1 info, 2 debug
- **seed** (*int / None*) – Seed for the pseudo random generators
- **device** (*device / str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance
- **use_sde** (*bool*) –
- **sde_sample_freq** (*int*) –

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = RecurrentPPO('MlpLstmPolicy', env, verbose = 1,
tensorboard_log=log_path)

model.learn(total_timesteps=20000)
#Saving the model
RecurrentPPO_path = os.path.join('Training', 'Saved
Models', 'RecurrentPPO_env')
model.save(RecurrentPPO_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

A2C

```
class stable_baselines3.a2c.A2C(policy, env, learning_rate=0.0007,
n_steps=5, gamma=0.99, gae_lambda=1.0, ent_coef=0.0, vf_coef=0.5,
```

```

max_grad_norm=0.5, rms_prop_eps=1e-05, use_rms_prop=True, use_sde=False,
sde_sample_freq=-1, rollout_buffer_class=None,
rollout_buffer_kwargs=None, normalize_advantage=False,
stats_window_size=100, tensorboard_log=None, policy_kwargs=None,
verbose=0, seed=None, device='auto', _init_setup_model=True)

```

Where the hyperparameters are described as follows.

- **policy** (*ActorCriticPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable[[float], float]*) – The learning rate, it can be a function of the current progress remaining (from 1 to 0)
- **n_steps** (*int*) – The number of steps to run for each environment per update (i.e. batch size is n_steps * n_env where n_env is number of environment copies running in parallel)
- **gamma** (*float*) – Discount factor
- **gae_lambda** (*float*) – Factor for trade-off of bias vs variance for Generalized Advantage Estimator. Equivalent to classic advantage when set to 1.
- **ent_coef** (*float*) – Entropy coefficient for the loss calculation
- **vf_coef** (*float*) – Value function coefficient for the loss calculation
- **max_grad_norm** (*float*) – The maximum value for the gradient clipping
- **rms_prop_eps** (*float*) – RMSProp epsilon. It stabilizes square root computation in denominator of RMSProp update
- **use_rms_prop** (*bool*) – Whether to use RMSprop (default) or Adam as optimizer
- **use_sde** (*bool*) – Whether to use generalized State Dependent Exploration (gSDE) instead of action noise exploration (default: False)
- **sde_sample_freq** (*int*) – Sample a new noise matrix every n steps when using gSDE Default: -1 (only sample at the beginning of the rollout)
- **rollout_buffer_class** (*Type[RolloutBuffer] | None*) – Rollout buffer class to use. If None, it will be automatically selected.
- **rollout_buffer_kwargs** (*Dict[str, Any] | None*) – Keyword arguments to pass to the rollout buffer on creation.
- **normalize_advantage** (*bool*) – Whether to normalize or not the advantage

- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over
- **tensorboard_log** (*str / None*) – the log location for tensorboard (if None, no logging)
- **policy_kwargs** (*Dict[str, Any] / None*) – additional arguments to be passed to the policy on creation
- **verbose** (*int*) – Verbosity level: 0 for no output, 1 for info messages (such as device or wrappers used), 2 for debug messages
- **seed** (*int / None*) – Seed for the pseudo random generators
- **device** (*device / str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = A2C('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
A2C_path = os.path.join('Training', 'Saved Models', 'A2C_env')
model.save(A2C_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

DDPG

```
class stable_baselines3.ddpg.DDPG(policy, env, learning_rate=0.001,
buffer_size=1000000, learning_starts=100, batch_size=100, tau=0.005,
gamma=0.99, train_freq=(1, 'episode'), gradient_steps=-1,
action_noise=None, replay_buffer_class=None, replay_buffer_kwargs=None,
optimize_memory_usage=False, tensorboard_log=None, policy_kwargs=None,
verbose=0, seed=None, device='auto', _init_setup_model=True)
```

Where the hyperparameters are described as follows.

- **policy** (*TD3Policy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)

- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable[[float], float]*) – learning rate for adam optimizer, the same learning rate will be used for all networks (Q-Values, Actor and Value function) it can be a function of the current progress remaining (from 1 to 0)
- **buffer_size** (*int*) – size of the replay buffer
- **learning_starts** (*int*) – how many steps of the model to collect transitions for before learning starts
- **batch_size** (*int*) – Minibatch size for each gradient update
- **tau** (*float*) – the soft update coefficient (“Polyak update”, between 0 and 1)
- **gamma** (*float*) – the discount factor
- **train_freq** (*int | Tuple[int, str]*) – Update the model every train_freq steps. Alternatively pass a tuple of frequency and unit like (5, "step") or (2, "episode").
- **gradient_steps** (*int*) – How many gradient steps to do after each rollout (see train_freq) Set to -1 means to do as many gradient steps as steps done in the environment during the rollout.
- **action_noise** (*ActionNoise | None*) – the action noise type (None by default), this can help for hard exploration problem. Cf common.noise for the different action noise type.
- **replay_buffer_class** (*Type[ReplayBuffer] | None*) – Replay buffer class to use (for instance HerReplayBuffer). If None, it will be automatically selected.
- **replay_buffer_kwargs** (*Dict[str, Any] | None*) – Keyword arguments to pass to the replay buffer on creation.
- **optimize_memory_usage** (*bool*) – Enable a memory efficient variant of the replay buffer at a cost of more complexity. See <https://github.com/DLR-RM/stable-baselines3/issues/37#issuecomment-637501195>
- **policy_kwargs** (*Dict[str, Any] | None*) – additional arguments to be passed to the policy on creation
- **verbose** (*int*) – Verbosity level: 0 for no output, 1 for info messages (such as device or wrappers used), 2 for debug messages
- **seed** (*int | None*) – Seed for the pseudo random generators
- **device** (*device | str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.

- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance
- **tensorboard_log** (*str / None*) –

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = DDPG('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
DDPG_path = os.path.join('Training', 'Saved Models', 'DDPG_env')
model.save(DDPG_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

SAC

```
class stable_baselines3.sac.SAC(policy, env, learning_rate=0.0003,
buffer_size=1000000, learning_starts=100, batch_size=256, tau=0.005,
gamma=0.99, train_freq=1, gradient_steps=1, action_noise=None,
replay_buffer_class=None, replay_buffer_kwargs=None,
optimize_memory_usage=False, ent_coef='auto', target_update_interval=1,
target_entropy='auto', use_sde=False, sde_sample_freq=-1,
use_sde_at_warmup=False, stats_window_size=100, tensorboard_log=None,
policy_kwargs=None, verbose=0, seed=None, device='auto',
_init_setup_model=True)
```

Where the hyperparameters are described as follows.

- **policy** (*SACPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env / VecEnv / str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float / Callable[[float], float]*) – learning rate for adam optimizer, the same learning rate will be used for all networks (Q-Values, Actor and Value function) it can be a function of the current progress remaining (from 1 to 0)
- **buffer_size** (*int*) – size of the replay buffer
- **learning_starts** (*int*) – how many steps of the model to collect transitions for before learning starts

- **batch_size** (*int*) – Minibatch size for each gradient update
- **tau** (*float*) – the soft update coefficient (“Polyak update”, between 0 and 1)
- **gamma** (*float*) – the discount factor
- **train_freq** (*int* / *Tuple[int, str]*) – Update the model every train_freq steps. Alternatively pass a tuple of frequency and unit like (5, "step") or (2, "episode").
- **gradient_steps** (*int*) – How many gradient steps to do after each rollout (see train_freq) Set to -1 means to do as many gradient steps as steps done in the environment during the rollout.
- **action_noise** (*ActionNoise* / *None*) – the action noise type (None by default), this can help for hard exploration problem. Cf common.noise for the different action noise type.
- **replay_buffer_class** (*Type[ReplayBuffer]* / *None*) – Replay buffer class to use (for instance HerReplayBuffer). If None, it will be automatically selected.
- **replay_buffer_kwargs** (*Dict[str, Any]* / *None*) – Keyword arguments to pass to the replay buffer on creation.
- **optimize_memory_usage** (*bool*) – Enable a memory efficient variant of the replay buffer at a cost of more complexity. See <https://github.com/DLR-RM/stable-baselines3/issues/37#issuecomment-637501195>
- **ent_coef** (*str* / *float*) – Entropy regularization coefficient. (Equivalent to inverse of reward scale in the original SAC paper.) Controlling exploration/exploitation trade-off. Set it to ‘auto’ to learn it automatically (and ‘auto_0.1’ for using 0.1 as initial value)
- **target_update_interval** (*int*) – update the target network every target_network_update_freq gradient steps.
- **target_entropy** (*str* / *float*) – target entropy when learning ent_coef (ent_coef = 'auto')
- **use_sde** (*bool*) – Whether to use generalized State Dependent Exploration (gSDE) instead of action noise exploration (default: False)
- **sde_sample_freq** (*int*) – Sample a new noise matrix every n steps when using gSDE Default: -1 (only sample at the beginning of the rollout)
- **use_sde_at_warmup** (*bool*) – Whether to use gSDE instead of uniform sampling during the warm up phase (before learning starts)
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over
- **tensorboard_log** (*str* / *None*) – the log location for tensorboard (if None, no logging)

- **policy_kwargs** (*Dict[str, Any] | None*) – additional arguments to be passed to the policy on creation
- **verbose** (*int*) – Verbosity level: 0 for no output, 1 for info messages (such as device or wrappers used), 2 for debug messages
- **seed** (*int | None*) – Seed for the pseudo random generators
- **device** (*device | str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = SAC('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
SAC_path = os.path.join('Training', 'Saved Models', 'SAC_env')
model.save(SAC_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

ARS

```
class sb3_contrib.ars.ARS(policy, env, n_delta=8, n_top=None,
learning_rate=0.02, delta_std=0.05, zero_policy=True,
alive_bonus_offset=0, n_eval_episodes=1, policy_kwargs=None,
stats_window_size=100, tensorboard_log=None, seed=None, verbose=0,
device='cpu', _init_setup_model=True)
```

Where the hyperparameters are described as follows.

- **policy** (*BasePolicy*) – The policy to train, can be an instance of ARSPolicy, or a string from ["LinearPolicy", "MlpPolicy"]
- **env** (*Env | VecEnv | str*) – The environment to train on, may be a string if registered with gym
- **n_delta** (*int*) – How many random perturbations of the policy to try at each update step.

- **n_top** (*int* / *None*) – How many of the top delta to use in each update step. Default is n_delta
- **learning_rate** (*float* / *Callable[[float], float]*) – Float or schedule for the step size
- **delta_std** (*float* / *Callable[[float], float]*) – Float or schedule for the exploration noise
- **zero_policy** (*bool*) – Boolean determining if the passed policy should have it's weights zeroed before training.
- **alive_bonus_offset** (*float*) – Constant added to the reward at each step, used to cancel out alive bonuses.
- **n_eval_episodes** (*int*) – Number of episodes to evaluate each candidate.
- **policy_kwargs** (*Dict[str, Any]* / *None*) – Keyword arguments to pass to the policy on creation
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over
- **tensorboard_log** (*str* / *None*) – String with the directory to put tensorboard logs:
- **seed** (*int* / *None*) – Random seed for the training
- **verbose** (*int*) – Verbosity level: 0 no output, 1 info, 2 debug
- **device** (*device* / *str*) – Torch device to use for training, defaults to “cpu”
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = ARS('LinearPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
ARS_path = os.path.join('Training', 'Saved Models', 'ARS_env')
model.save(ARS_path)
#Evaluating the model performance over 50 episodes
evaluate_policy(model, env_run, n_eval_episodes=50)
```

TQC

```
class sb3_contrib.tqc.TQC(policy, env, Learning_rate=0.0003,
buffer_size=1000000, Learning_starts=100, batch_size=256, tau=0.005,
```



```

gamma=0.99, train_freq=1, gradient_steps=1, action_noise=None,
replay_buffer_class=None, replay_buffer_kwargs=None,
optimize_memory_usage=False, ent_coef='auto', target_update_interval=1,
target_entropy='auto', top_quantiles_to_drop_per_net=2, use_sde=False,
sde_sample_freq=-1, use_sde_at_warmup=False, stats_window_size=100,
tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None,
device='auto', _init_setup_model=True)

```

Where the hyperparameters are described as follows.

- **policy** (*TQCPolicy*) – The policy model to use (MlpPolicy, CnnPolicy, ...)
- **env** (*Env | VecEnv | str*) – The environment to learn from (if registered in Gym, can be str)
- **learning_rate** (*float | Callable*) – learning rate for adam optimizer, the same learning rate will be used for all networks (Q-Values, Actor and Value function) it can be a function of the current progress remaining (from 1 to 0)
- **buffer_size** (*int*) – size of the replay buffer
- **learning_starts** (*int*) – how many steps of the model to collect transitions for before learning starts
- **batch_size** (*int*) – Minibatch size for each gradient update
- **tau** (*float*) – the soft update coefficient (“Polyak update”, between 0 and 1)
- **gamma** (*float*) – the discount factor
- **train_freq** (*int*) – Update the model every train_freq steps. Alternatively pass a tuple of frequency and unit like (5, "step") or (2, "episode").
- **gradient_steps** (*int*) – How many gradient update after each step
- **action_noise** (*ActionNoise | None*) – the action noise type (None by default), this can help for hard exploration problem. Cf common.noise for the different action noise type.
- **replay_buffer_class** (*Type[ReplayBuffer] | None*) – Replay buffer class to use (for instance HerReplayBuffer). If None, it will be automatically selected.
- **replay_buffer_kwargs** (*Dict[str, Any] | None*) – Keyword arguments to pass to the replay buffer on creation.
- **optimize_memory_usage** (*bool*) – Enable a memory efficient variant of the replay buffer at a cost of more complexity. See <https://github.com/DLR-RM/stable-baselines3/issues/37#issuecomment-637501195>

- **ent_coef** (*str / float*) – Entropy regularization coefficient. (Equivalent to inverse of reward scale in the original SAC paper.) Controlling exploration/exploitation trade-off. Set it to 'auto' to learn it automatically (and 'auto_0.1' for using 0.1 as initial value)
- **target_update_interval** (*int*) – update the target network every target_network_update_freq gradient steps.
- **target_entropy** (*str / float*) – target entropy when learning ent_coef (ent_coef = 'auto')
- **top_quantiles_to_drop_per_net** (*int*) – Number of quantiles to drop per network
- **use_sde** (*bool*) – Whether to use generalized State Dependent Exploration (gSDE) instead of action noise exploration (default: False)
- **sde_sample_freq** (*int*) – Sample a new noise matrix every n steps when using gSDE Default: -1 (only sample at the beginning of the rollout)
- **use_sde_at_warmup** (*bool*) – Whether to use gSDE instead of uniform sampling during the warm up phase (before learning starts)
- **stats_window_size** (*int*) – Window size for the rollout logging, specifying the number of episodes to average the reported success rate, mean episode length, and mean reward over
- **tensorboard_log** (*str / None*) – the log location for tensorboard (if None, no logging)
- **policy_kwargs** (*Dict[str, Any] / None*) – additional arguments to be passed to the policy on creation
- **verbose** (*int*) – the verbosity level: 0 no output, 1 info, 2 debug
- **seed** (*int / None*) – Seed for the pseudo random generators
- **device** (*device / str*) – Device (cpu, cuda, ...) on which the code should be run. Setting it to auto, the code will be run on the GPU if possible.
- **_init_setup_model** (*bool*) – Whether or not to build the network at the creation of the instance

The training, saving and evaluation of the model is shown below.

```
#Creating and training the models
log_path = os.path.join('Training', 'Logs')
model = TQC('MlpPolicy', env, verbose = 1, tensorboard_log=log_path)
model.learn(total_timesteps=20000)
#Saving the model
TQC_path = os.path.join('Training', 'Saved Models', 'TQC_env')
model.save(TQC_path)
#Evaluating the model performance over 50 episodes
```

```
evaluate_policy(model, env_run, n_eval_episodes=50)
```

Appendix D: Testing Blocks (DRL and non-DRL)

In this appendix, we highlight the code used to test the DRL and non-DRL algorithms. This is done through semi-identical testing blocks shown below. The only difference is in the function used to call the algorithm's action and the outputs of the function.

Non-DRL algorithms

```
#Select optimisation algorithm class
opt = pso()

for h in range(3): #Choose the no. of episodes for the test
    env.reset()
    env.episode = h
    episodic_violations = 0

    while not env.done:
        print(f'Observation:
{env.observation}')

        result, action = opt.maximise(env.step_count)
        print(result)
        obs, reward, done, info = env.step(action)
        print(f'action: {action}')
        episodic_violations += env.action_df['vio_after'].sum()
        #print('observation: {}'.format(obs))
        print(f'step: {env.step_count} \nepisode: {env.episode}')
        print('reward: {}'.format(reward))
        print('done: {}'.format(done))
        print(f'state: {env.state}')
        env.report('excel') #report step logs to excel

    print(f'State logs: \n {env.state_logs1}')
    #Create data visualisation figures
    env.render('interactive map')
    env.render('settings')
    env.render('states')
    env.render('reward across junctions')
    env.render('water loss')
```

DRL Algorithms

```
#Load saved trained model from path e.g. Recurrent PPO
path = os.path.join('Training', 'SZ08 Models', 'RecurrentPPO_SZ08')
model = RecurrentPPO.load(path)

for h in range(3): #Choose the no. of episodes for the test
    env.reset()
    env.episode = h
    episodic_violations = 0

    while not env.done:
        obs = env.observation
        print(f'Observation:
{env.observation}')

        action = model.predict(obs)
        print(f'action: {action}')
        obs, reward, done, info = env.step(action[0])
        episodic_violations += env.action_df['vio_after'].sum()
        #print('observation: {}'.format(obs))
        print(f'step: {env.step_count} \nepisode: {env.episode}')
        print('reward: {}'.format(reward))
        print('done: {}'.format(done))
        print(f'state: {env.state}')
        env.report('excel')

    print(f'State logs: \n {env.state_logs1}')
    #Create data visualisation figures
    env.render('interactive map')
    env.render('settings')
    env.render('states')
    env.render('reward across junctions')
    env.render('water loss')
```

Appendix E: Reward Scales Sweep

In this appendix, we detail the code used to test different reward scales as mentioned in section 3.2.4.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from WaterNetwork import WaterNetwork
from stable_baselines3 import A2C

#Leakage junctions uid depends on the test scenario and water networks
used
leak_junc = ['N10089457', 'NX33952610', 'RX33640557', 'N16111134',
             'N12717876', 'R10204722', 'N10091679', 'N12717855', 'N10162466',
             'LX33640555', 'N12717878', 'N10204302',
             'N10210130', 'NX33876614', 'N16111934', 'N13656815',
             'N10089472', 'N10258111', 'N13663122', 'N10204291', 'R13656833',
             'N10162944', 'N10210150', 'N16111209',
             'N10355184', 'N16111527', 'N10091659', 'N12707260',
             'N10091649', 'NX34272347', 'N10209737', 'N10296836']

env = WaterNetwork(wdn_name='SZ08_PEAK', burst=True,
                  reset_orig_junc=True, orig_junc=leak_junc, n_junc=32, scale=[1,1])

A2C11_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-11')
A2C12_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-12')
A2C13_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-13')
A2C14_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-14')
A2C15_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-15')
A2C16_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-16')
A2C17_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-17')
A2C21_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-21')
A2C31_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-31')
A2C41_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-41')
A2C51_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-51')
A2C61_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-61')
A2C71_path = os.path.join('Training', 'SZ08 Models', 'A2C_SZ08-71')
```

```

model11 = A2C.load(A2C11_path)
model12 = A2C.load(A2C12_path)
model13 = A2C.load(A2C13_path)
model14 = A2C.load(A2C14_path)
model15 = A2C.load(A2C15_path)
model16 = A2C.load(A2C16_path)
model17 = A2C.load(A2C17_path)
model21 = A2C.load(A2C21_path)
model31 = A2C.load(A2C31_path)
model41 = A2C.load(A2C41_path)
model51 = A2C.load(A2C51_path)
model61 = A2C.load(A2C61_path)
model71 = A2C.load(A2C71_path)

model_list = [model71, model61, model51, model41, model31, model21,
model11, model12, model13, model14, model15, model16, model17]
model_names = [7,6,5,4,3,2,1,-2,-3,-4,-5,-6,-7]
modelnames =
['7:1', '6:1', '5:1', '4:1', '3:1', '2:1', '1:1', '1:2', '1:3', '1:4', '1:5', '1:6',
'1:7']
times = np.arange(0,24)
lr_logs = np.empty((24, len(model_list)))
vio_logs = np.empty((24, len(model_names)))

for m in model_list:
    env.reset()
    while not env.done:
        print(f'Step: {env.step_count}')
        obs = env.observation
        print(f'Observation:
{env.observation}')

        action = m.predict(obs)
        print(f'action: {action}')
        obs, reward, done, info = env.step(action[0])

```

```

        lr_logs[env.step_count, model_list.index(m)] =
env.action_df['leak_after'].sum()
        vio_logs[env.step_count, model_list.index(m)] =
env.action_df['vio_after'].sum()
penalty = np.add(lr_logs, vio_logs)

sum_penalty = np.sum(penalty,axis=0)
print(f'Best Ratio: {modelnames[sum_penalty.argmin()]}')

for m in range(len(model_list)):
    plt.plot(times, penalty[:, m], label=f'{model_names[m]}')
plt.xlabel('Times')
plt.ylabel('Penalty')
plt.legend()
plt.show()

plt.bar(modelnames, np.sum(penalty, axis=0))
plt.xlabel('Reward scale ratios')
plt.ylabel('Penalty')
plt.yscale('log')
plt.title('Penalty per episode for reward ratios')
plt.show()

plt.bar(modelnames, np.sum(lr_logs, axis=0))
plt.xlabel('Reward scale ratios')
plt.ylabel('Leakage')
plt.yscale('log')
plt.title('Leak per episode for reward ratios')
plt.show()

plt.bar(modelnames, np.sum(vio_logs, axis=0))
plt.xlabel('Reward scale ratios')
plt.ylabel('Violations')
plt.title('Violations per episode for reward ratios')
plt.show()

```



```
x, y = np.meshgrid(model_names, times)
z = lr_logs
ax = plt.axes(projection = "3d", xlabel = "Reward Scales", ylabel =
"time", zlabel = "Leakage rates")
ax.plot_surface(x,y,z, cmap='plasma')
plt.show()
```

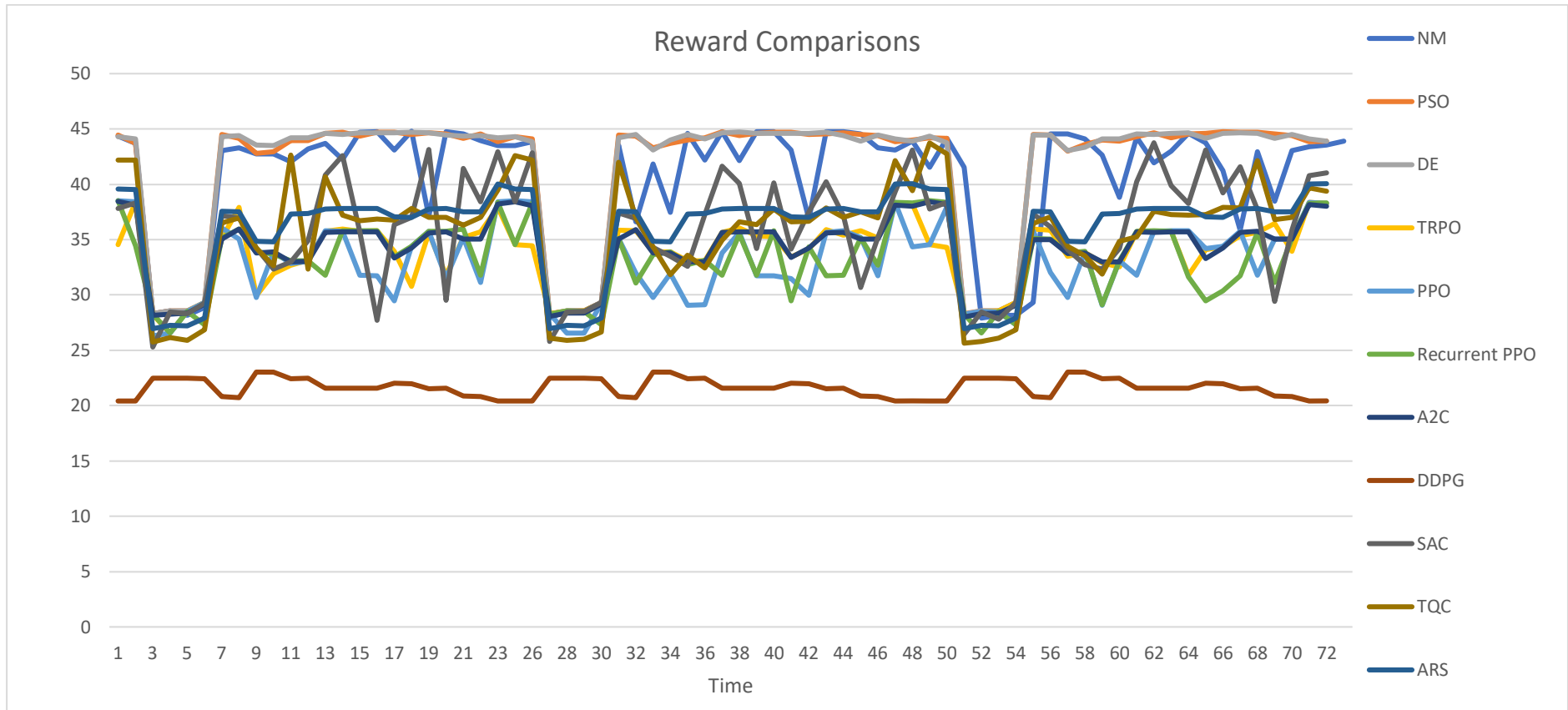
```
x, y = np.meshgrid(model_names, times)
z = vio_logs
ax = plt.axes(projection = "3d", xlabel = "Reward Scales", ylabel =
"time", zlabel = "Violations")
ax.plot_surface(x,y,z, cmap = 'plasma')
plt.show()
```

```
x, y = np.meshgrid(model_names, times)
z = penalty
ax = plt.axes(projection = "3d", xlabel = "Reward Scales", ylabel =
"time", zlabel = "Penalty")
ax.plot_surface(x,y,z, cmap = 'plasma')
plt.show()
```

Appendix F: Background Leakage – Jowitt & Xu Results

This section shows the episodic performance of each of the optimisation algorithms as they tackle the background leakage case study on the Jowitt & Xu network. The results and discussions associated with these results were covered in section 4.2 of the thesis.

Each algorithm's step rewards, water saved% and carbon emissions are listed along with the algorithm's processing speeds. A line plot of the rewards of the algorithms across the three test episodes is shown below.



NM

NM	Time Elapse	233.3
----	-------------	-------

Step	Episode 0				Episode 1				Episode 2			
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	19.61539081	69.885809	44.33126629	12.4777424	19.8528557	70.73185005	43.47437162	12.62879857	19.13852888	68.186843	44.14838205	12.17440099
1	20.4059058	72.57650629	43.70804933	12.9806048	19.00332322	67.58802184	43.84385154	12.08839397	19.90810741	70.80601551	41.5232781	12.66394529
2	13.83142452	47.91126025	28.30332017	8.798445766	12.64853445	43.81379699	27.90089107	8.045985737	12.64853499	43.81379884	27.90089229	8.045986077
3	13.94184389	48.30922661	28.5662216	8.868685733	13.94176281	48.30894569	28.56643534	8.868634162	12.75465265	44.19554618	28.15381584	8.113489646
4	12.75440652	44.19466049	28.15322891	8.113333077	13.94158695	48.30830042	28.5660093	8.86852229	12.75433489	44.19441227	28.15306379	8.113287507
5	13.05620705	45.28200086	28.87382743	8.305314428	14.25589861	49.44281372	29.3168258	9.068462222	14.25591291	49.44286334	29.31685671	9.068471323
6	19.34953974	71.1063847	43.02716265	12.30862922	19.63723475	72.16361669	43.53205018	12.49163777	19.09254452	70.16196943	44.55767371	12.14514942
7	18.20455109	66.84373171	43.31680432	11.58027904	20.34711357	74.71082337	36.62361457	12.94320589	19.09935733	70.12929411	44.53202691	12.14948319
8	16.95517485	67.23640694	42.75030508	10.78552582	17.50041528	69.39857914	41.85104179	11.13236417	17.43748458	69.149025	44.10751583	11.09233269
9	16.94640947	67.22983488	42.74693987	10.77994999	18.39068252	72.95955824	37.46701328	11.69868096	17.69705541	70.2077992	42.63059718	11.25745089
10	18.09828098	70.05762106	42.05055061	11.5126785	18.0795098	69.98495865	44.61387331	11.50073777	19.5051474	75.50353685	38.82756552	12.40761436
11	18.43377706	71.41023153	43.20700764	11.72609426	18.14202834	70.28003214	42.2027164	11.54050707	17.91722074	69.40915461	44.23157307	11.39750246
12	18.38708941	68.25489122	43.69274211	11.69639532	18.96869682	70.41388166	44.7638486	12.06636742	18.65661398	69.25539595	41.95280092	11.86784528
13	18.54931357	68.85431152	42.24569225	11.79958935	17.75962529	65.92302017	42.12595455	11.29725284	17.86654525	66.31990276	42.96952135	11.36526676
14	18.94554976	70.34806828	44.71825769	12.05164311	18.96349056	70.41468554	44.76509091	12.06305562	18.87544998	70.08777579	44.60259148	12.00705124
15	18.96924858	70.41490445	44.76424857	12.06671841	18.96927287	70.4149946	44.76431079	12.06673386	18.42878037	68.40865643	43.76917326	11.72291577
16	17.89676617	67.92631334	43.10980616	11.38449089	17.90041451	67.94016045	43.11647531	11.38681168	16.62066866	63.08294675	41.24044891	10.57273975
17	18.58189079	70.36620689	44.79986767	11.82031237	19.76261915	74.83740831	36.8399336	12.5713973	20.75671902	78.60188189	35.86896689	13.2037641
18	20.33337364	75.37414404	37.11418139	12.93446564	18.99380729	70.40848172	44.75712115	12.08234069	17.87050955	66.24450938	42.94167802	11.36778853
19	18.95149306	70.37381183	44.73766587	12.05542376	18.95884342	70.40110639	44.75619581	12.06009947	19.84177948	73.67977029	38.47685121	12.62175276
20	19.08761214	70.1871777	44.57718114	12.14201183	19.08743571	70.18652895	44.57675544	12.1418996	19.34680373	71.14025272	43.0503535	12.30688879
21	18.74777909	68.89814352	43.92490507	11.92583723	18.59062981	68.32061949	43.28747462	11.82587144	18.22259304	66.96808327	43.37585432	11.59175589
22	19.90491999	70.7935195	43.51025591	12.66191771	20.55926272	73.12074435	43.07233733	13.0781582	19.90863296	70.80672496	43.52373516	12.6642796
23			28.48268682	11.27791686	19.04166934	67.71557759	43.91080564	11.38353457	19.0416682	67.71557354	43.91080357	11.23894921
Total	409.947948	65.81892029	964.7125751	272.0540055	433.2967135	66.57452109	968.6949979	274.8994533	427.6456459	65.7296555	963.7660196	271.1601115

PSO

PSO												
		Time Elapsed		1274.1								
Step	Episode 0		Episode 0	Episode 0	Episode 1		Episode 1	Episode 1	Episode 2		Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	19.66995625	70.08021502	44.42858419	12.51245257	19.61105514	69.87036186	44.32206527	12.4749844	19.56387935	69.70228372	44.2085957	12.44497493
1	20.40446777	72.57139171	43.7157519	12.97969004	19.15162567	68.1154806	44.11229311	12.18273212	19.16289468	68.15556045	44.13765505	12.18990057
2	13.83138509	47.91112365	28.30323498	8.798420681	13.83123414	47.91060078	28.30291083	8.798324659	13.83127943	47.91075765	28.30300797	8.798353468
3	13.94166408	48.30860358	28.56623507	8.868571356	13.94170386	48.3087414	28.56630831	8.868596658	13.94165331	48.30856624	28.56621211	8.868564502
4	13.94155517	48.30819031	28.56594085	8.868502076	13.94152963	48.30810179	28.56588603	8.868485826	13.94157918	48.30827349	28.56599275	8.868517347
5	14.25588711	49.44277384	29.31680112	9.068454907	14.25541449	49.44113469	29.31578523	9.068154265	14.25573878	49.44225939	29.31646774	9.068360551
6	19.06854642	70.07378037	44.49909016	12.12988375	19.04440735	69.98507323	44.46689529	12.1145284	19.05214693	70.01351494	44.47805102	12.11945171
7	18.89327031	69.37258084	44.14780269	12.01838711	18.99266678	69.7375462	44.33006353	12.08161519	19.05921104	69.98188437	44.45189819	12.12394533
8	16.88722751	66.96695911	42.80004212	10.74230316	17.09371697	67.78580111	43.27545218	10.87365524	17.00846559	67.44773346	43.00516464	10.81942513
9	16.98152488	67.36914482	42.95795413	10.80228761	17.26456059	68.49200472	43.67789345	10.98233228	17.20116549	68.24050356	43.58797692	10.94200539
10	17.81000529	68.94171896	43.93005747	11.32930056	17.79468012	68.88239596	43.98897583	11.31955192	17.78096378	68.8293006	44.00214364	11.31082668
11	17.73329193	68.69663655	43.96020026	11.28050166	17.90430622	69.35912529	44.20727823	11.38928727	17.77162873	68.84514867	43.91631861	11.30488847
12	18.87914009	70.08143728	44.58244328	12.0093986	18.95644531	70.36840273	44.74106918	12.05857399	18.75514962	69.62117106	44.30373559	11.93052578
13	18.93739399	70.29485056	44.70456257	12.04645507	18.78131905	69.71550659	44.40858411	11.94717267	18.91237941	70.20199744	44.65555272	12.03054279
14	18.73531649	69.56743615	44.3437475	11.91790952	18.89130929	70.14666411	44.62267543	12.01713966	18.66079276	69.29071682	44.20750015	11.87050349
15	18.94081234	70.30934755	44.69646909	12.04862954	18.9374252	70.2967743	44.69242006	12.04647492	18.88174399	70.09008257	44.56106441	12.01105499
16	18.48228328	70.1486153	44.70163847	11.75695004	18.47701786	70.12863069	44.68588379	11.7536006	18.4358726	69.97246584	44.6152822	11.72742728
17	18.43378043	69.80534017	44.50340952	11.72609641	18.4207271	69.75590962	44.49628795	11.71779292	18.55553952	70.2664195	44.73436492	11.8035498
18	18.95017923	70.24675609	44.67315216	12.05458801	18.89545625	70.04390251	44.57229596	12.01977763	18.97626165	70.34344151	44.7242883	12.07117956
19	18.87553917	70.09176729	44.57487029	12.00710797	18.90747037	70.21033951	44.64182324	12.02742005	18.9339779	70.30877166	44.70717861	12.04428202
20	18.91081765	69.53708558	44.13925863	12.02954932	19.05697048	70.07450504	44.51313106	12.12252006	19.06809416	70.115408	44.52995581	12.12959606
21	19.08331119	70.13122499	44.54063516	12.13927591	19.00282569	69.83544054	44.39244631	12.08807748	19.0182246	69.89203158	44.40840306	12.09787303
22	19.0077599	67.60269428	43.85143438	12.09121623	18.99889421	67.57116272	43.832679	12.08557659	19.00721941	67.60077198	43.85029146	12.09087241

23

Total

28.48268682	11.32430527	19.09840331	67.91733373	44.01336194	11.31003628	19.03758577	67.7010557	43.90207085	11.29889768		
410.6551156	65.90694235	982.9860028	272.5502374	429.2511651	65.92753916	1000.744465	272.2164111	428.8134477	65.85792167	999.7391724	271.965519

DE

DE		Time Elapsed		676									
Step		Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
		Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0		19.61135434	69.87142783	44.322701	12.47517472	19.60965027	69.86535658	44.31908499	12.47409073	19.61531405	69.8855355	44.33110435	12.47769357
1		19.15186231	68.11632223	44.10339914	12.18288265	19.05631785	67.77650478	43.87275048	12.12210491	18.96267525	67.44345155	43.75792411	12.06253698
2		13.82334242	47.88326431	28.28632337	8.793304578	13.81957053	47.87019872	28.2784661	8.790905207	13.82035171	47.87290469	28.28009159	8.791402132
3		13.93817374	48.29650935	28.55885156	8.866351078	13.93739322	48.29380482	28.55721255	8.865854575	13.93930186	48.30041835	28.5612259	8.867068698
4		13.93992351	48.30253652	28.56246819	8.867464144	13.94091868	48.30598484	28.56457907	8.868097192	13.94117649	48.30687817	28.56512849	8.868261191
5		14.24739644	49.4133262	29.29890245	9.063053822	14.25128608	49.42681638	29.30705597	9.0655281	14.24669724	49.41090122	29.29744087	9.06260905
6		18.95108387	69.64212476	44.27784136	12.05516347	18.93538022	69.58441643	44.20518835	12.04517407	19.03441024	69.94833548	44.44697185	12.10816904
7		19.04583283	69.93276208	44.41863233	12.11543518	19.08723158	70.08477059	44.50703142	12.14176975	19.06071233	69.98739683	44.43759706	12.12490033
8		17.22948555	68.32419674	43.55891047	10.96002035	17.01258476	67.46406819	43.07593191	10.82204542	17.97213024	71.26918322	43.01996025	11.43243149
9		17.17610667	68.14109011	43.49022321	10.92606497	17.38048156	68.95188666	43.9836838	11.05607193	17.15937499	68.07471219	43.36840208	10.91542162
10		17.91246248	69.33832609	44.1962883	11.39447563	18.03717622	69.82108731	44.50105834	11.47380854	17.87541369	69.19491191	44.1088283	11.37090816
11		17.90470071	69.36065352	44.2106846	11.38953822	17.86838788	69.21998199	44.10296135	11.3664389	17.87493457	69.24534308	44.1255376	11.37060338
12		18.8928284	70.13224977	44.6115277	12.018106	18.92122008	70.23764281	44.65181146	12.03616652	18.87045125	70.04918334	44.57327875	12.00387145
13		18.85133935	69.97541918	44.52446567	11.99171399	18.95502087	70.36028081	44.72821724	12.05766787	18.8420635	69.94098759	44.4947381	11.98581344
14		18.90463193	70.19613339	44.65650131	12.02561447	18.90451044	70.19568225	44.62562923	12.02553718	18.88369298	70.11838348	44.59041154	12.01229478
15		18.92247932	70.24129435	44.67389986	12.03696754	18.89822273	70.15125255	44.58762813	12.02153744	18.92038847	70.23353302	44.65312639	12.03563752
16		18.46538259	70.08446957	44.64494861	11.74619917	18.43162131	69.95633026	44.60300239	11.72472295	18.19901001	69.07346527	44.15787052	11.57675425
17		18.5431389	70.21946062	44.72616777	11.79566152	18.4870887	70.0072088	44.58727987	11.76000686	18.4927008	70.02846081	44.58972081	11.76357684
18		18.94343971	70.22177324	44.64244237	12.05030087	18.96738976	70.31055411	44.70472868	12.06553597	18.93811632	70.20203988	44.65250014	12.04691455

19	18.82563934	69.90647101	44.472743	11.97536569	18.79667234	69.79890598	44.4059638	11.95693921	18.89519804	70.16476787	44.61715075	12.01961338
20	19.00022002	69.86582757	44.3754479	12.08641996	18.73867417	68.90409569	43.87935712	11.92004541	18.90000885	69.49734047	44.16793515	12.02267363
21	19.02811832	69.92839103	44.41204934	12.10416662	19.04872136	70.00410727	44.45691992	12.11727263	19.06480796	70.06322559	44.50485763	12.12750564
22	19.1853491	68.23430519	44.17962229	12.20418427	19.14062686	68.07524678	44.09143046	12.17573556	19.13853939	68.06782251	44.09183405	12.17440767
23			28.48268682	11.31452021	19.03325314	67.68564809	43.89296345	11.30806006	19.04055742	67.71162339	43.90844818	11.32030278
Total	410.4942918	65.89688412	985.6877286	272.4381491	429.2594006	65.93132636	1000.489936	272.261117	429.6880277	66.00378356	999.3020845	272.5413716

TRPO

TRPO	Time Elapsed (training) 259				Time Elapsed (testing) 27							
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
Step	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	21.55316484	76.78971964	34.53202962	13.71039922	21.55316484	76.78971964	34.53202962	13.71039922	21.55316484	76.78971964	34.53202962	13.71039922
1	21.54185874	76.61668448	38.41515944	13.70320718	21.55247349	76.65443735	34.43099757	13.70995944	21.49045498	76.43385968	34.29128725	13.67050822
2	13.78043662	47.73464108	28.19765035	8.76601134	13.82637237	47.8937599	28.29264039	8.795231993	13.79760145	47.79409909	28.23298241	8.776930236
3	13.94176232	48.30894398	26.56643433	8.868633847	13.9236389	48.24614539	28.52853726	8.85710518	13.90048725	48.16592366	28.48070274	8.842377947
4	13.926871	48.2573089	28.5351678	8.859161179	13.94092796	48.30601698	28.56458512	8.868103093	13.93010541	48.26851631	28.54189479	8.861218653
5	14.25053402	49.42420807	29.30545827	9.0650497	14.224198	49.33286854	29.25068622	9.04829683	14.25201806	49.42935505	29.30857828	9.065993725
6	21.08792736	77.4946741	35.07580183	13.41445236	21.0039459	77.186056	35.85299269	13.36103007	21.04318653	77.33025886	35.95546042	13.38599181
7	20.64761404	75.81420531	37.93998804	13.13436025	20.99701487	77.09714028	35.7913995	13.3566211	21.0231902	77.19325122	35.85448183	13.37327175
8	20.50874139	81.32821362	29.90754893	13.04602057	20.4933313	81.2671043	33.87529929	13.03621791	20.351182	80.70340571	33.48937221	12.94579389
9	20.50421495	81.344369	31.92029665	13.04314121	20.49073229	81.29088054	33.89240499	13.03456462	20.47898572	81.24427953	33.8609412	13.02709239
10	20.57928791	79.66148583	32.6751987	13.09089663	20.73630965	80.26930983	33.09228462	13.1907813	20.65075258	79.93812231	32.87459598	13.13635673
11	20.72871782	80.30055556	33.11646307	13.18595198	20.71075878	80.23098441	33.08085224	13.17452788	20.51041271	79.45486788	32.53543532	13.04708374
12	21.13464319	78.45411199	35.74738725	13.44416923	21.10129925	78.33033561	35.65252416	13.42295848	21.12929589	78.43426223	35.72561457	13.4407677
13	20.82100613	77.28674363	35.94885425	13.24465842	20.86900095	77.4648985	36.06853194	13.27518889	21.11995104	78.3964152	35.71717437	13.43482326

14	21.12790397	78.45152291	35.74851529	13.43988227	20.96797905	77.85769433	35.33882228	13.33815084	21.13065476	78.46173705	35.75380519	13.4416321
15	21.15199303	78.5173731	35.78014356	13.45520581	20.93668149	77.71812471	35.24179553	13.31824183	21.15833213	78.54090421	31.79289058	13.45923824
16	20.72888401	78.67548011	33.95424721	13.1860577	20.91800881	79.39329418	33.43825498	13.30636377	20.78278097	78.8800434	34.08436445	13.22034263
17	20.72183251	78.46977309	30.79473208	13.1815721	20.85258883	78.96492324	34.13937803	13.2647488	20.94153545	79.30174777	34.36873826	13.32132953
18	21.10794678	78.24542296	35.59632439	13.4271871	20.83740087	77.24253153	35.92543439	13.25508744	21.01389097	77.89676582	35.36579409	13.36735632
19	21.1545999	78.55475176	31.80310469	13.45686409	21.00009934	77.98103478	35.4151686	13.35858319	21.09882769	78.34764918	35.66627886	13.42138627
20	21.12106375	77.6643953	35.1771513	13.43553107	20.97324253	77.12084095	35.81388662	13.34149904	20.83331004	76.60629433	36.46324604	13.25248518
21	20.93273821	76.92787476	35.68531295	13.31573343	21.11245542	77.58833603	35.12545487	13.43005514	21.03030923	77.28644855	33.91919677	13.37780031
22	21.33825116	75.89128217	37.9291457	13.57368833	21.49067117	76.43337674	38.30079513	13.67064575	21.52484623	76.55492318	38.37293588	13.69238518
23	17.31404686			12.52155212	21.46500425	76.33338942	38.22511495	12.52062346	21.47280134	76.36111726	38.25061592	12.52637181
Total	471.7060405	73.05277136	770.3521157	301.5693871	475.9773003	73.20805013	817.877971	301.6449852	476.2180775	73.24224863	813.438417	301.7989369

PPO

PPO	Time Elapsed (train) 216				Time Elapsed (Test) 26.6							
Step	Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	21.54644857	76.76579089	38.51931049	13.70612686	21.53834229	76.73690979	38.50412645	13.7009703	21.55316484	76.78971965	34.53202962	13.71039922
1	21.54912142	76.64251523	38.43138845	13.70782712	21.54347374	76.62242846	38.41875512	13.70423452	21.35109597	75.93820955	37.96177756	13.58185917
2	13.83134433	47.91098248	26.30313557	8.798394756	13.82880785	47.90219624	28.2977557	8.796781247	13.81273835	47.84653242	28.26426583	8.786559117
3	13.94176241	48.30894431	26.56643456	8.868633907	13.9417628	48.30894564	26.56643531	8.868634152	13.91938969	48.23142162	28.51973193	8.854402167
4	13.92653078	48.25613004	28.53446081	8.858944762	13.94150735	48.30802462	26.56582591	8.868471658	13.89452759	48.14523738	28.46834818	8.838586892
5	14.24746601	49.41356749	29.29902932	9.063098078	14.21327454	49.29498342	29.22811821	9.041348201	14.25582682	49.44256474	27.31665815	9.068416555
6	21.08007485	77.4658174	36.04411603	13.40945722	21.10419104	77.55444044	35.10683189	13.424798	20.9907852	77.13769256	35.82752995	13.35265828
7	21.0984446	77.46957142	35.045671	13.42114258	21.07031549	77.36628655	31.97045339	13.40324909	21.08994689	77.43836939	32.01911455	13.41573701
8	20.45297136	81.10705538	29.75464075	13.01054414	20.45501065	81.11514227	29.76023097	13.01184137	20.45415096	81.11173314	29.75787434	13.01129451
9	20.49646188	81.31361102	33.90398833	13.03820933	20.50406978	81.3437931	31.91986155	13.04304887	20.4988022	81.32289555	33.90874913	13.03969806

10	20.64091209	79.9000302	32.83921426	13.130097	20.72536741	80.22695291	29.0612108	13.18382072	20.73223698	80.2535447	29.07944978	13.18819059
11	20.69142601	80.1560916	33.01659713	13.16222992	20.73326675	80.31817759	29.12631905	13.18884565	20.71700553	80.25518355	33.09318576	13.17850156
12	21.15692181	78.53681265	35.79049818	13.4583411	21.15791687	78.54050643	33.79252872	13.45897408	21.15817798	78.54147569	31.79331205	13.45914017
13	21.15080109	78.51092934	35.77656101	13.45444759	21.098287	78.31599914	35.64631683	13.42104233	21.1481277	78.50100583	35.76728221	13.45274699
14	21.15515216	78.55270009	31.8015912	13.45721539	21.13663492	78.48394242	31.75501595	13.44543621	21.14851416	78.52805205	35.78825976	13.45299282
15	21.12509131	78.41751242	31.70931575	13.43809308	21.12372287	78.41243268	31.70587534	13.43722259	21.15124058	78.51457998	35.77868413	13.45472716
16	20.9320127	79.44644526	29.4663661	13.31527192	20.93167936	79.44518007	31.46537659	13.31505987	20.82447887	79.03830577	34.18816924	13.2468675
17	20.95181913	79.34069017	34.38893292	13.32787118	20.79788177	78.75775766	29.99079818	13.22994855	20.94929712	79.33113976	34.38394431	13.32626688
18	21.17026703	78.47643903	35.74716181	13.46683026	21.12317621	78.30187721	35.62961863	13.43687485	21.16347939	78.45127778	35.73398769	13.46251251
19	21.03618911	78.11504926	31.5052927	13.38154062	21.1528969	78.54842789	35.79940334	13.45578078	21.1545999	78.55475175	31.80310468	13.45686409
20	21.12134827	77.66544151	35.17770363	13.43571206	21.12157858	77.66628838	35.17815145	13.43585857	21.07212122	77.48442841	35.05939645	13.40439775
21	21.11728366	77.60607984	31.13504227	13.43312648	20.96087989	77.0312955	31.74696744	13.33363491	21.07876084	77.46450836	35.05445592	13.40862135
22	21.54533716	76.6278009	38.42150352	13.70541987	21.52071013	76.5402128	38.36683596	13.68975413	21.52473746	76.55453635	38.37572539	13.69231599
23				12.56105252	21.51163311	76.49920998	34.33491425	12.55481257	21.49074836	76.42494008	38.2863199	12.5523382
Total	455.9651877	73.30460904	759.1779558	302.6096277	477.2363873	73.40172547	779.937727	302.4504432	477.1339546	73.38758775	800.7613565	302.3960945

Recurrent PPO

Recurrent PPO		Time Elapsed (train) 401				Time Elapsed (test) 27						
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards		Water Saved	Water Saved %	Rewards		Water Saved	Water Saved %	Rewards	
0	21.54056294	76.74482155	38.5082841	13.7023829	21.55316484	76.78971965	34.53202962	13.71039922	21.54885089	76.77434989	38.52381673	13.70765503
1	21.5524735	76.65443739	34.4390976	13.70995945	21.50752036	76.49455513	38.33744355	13.68136385	21.52664073	76.56255947	38.38145522	13.6935267
2	13.82122434	47.8759274	28.28189101	8.791957226	13.8209465	47.874965	28.28131202	8.791780489	13.80061585	47.8045408	28.23920132	8.778847757
3	13.94176287	48.3089459	26.56643552	8.868634199	13.93669762	48.29139451	28.55573693	8.865412088	13.94176232	48.30894398	26.56643436	8.868633848
4	13.9261868	48.25493813	28.5337463	8.858725948	13.91226286	48.20669096	28.50491146	8.849868647	13.92089847	48.23661379	28.52277653	8.855361933
5	14.25582678	49.44256461	27.31665806	9.068416532	14.25582685	49.44256484	27.31665821	9.068416573	14.25582676	49.44256454	27.31665802	9.068416519

6	21.11569719	77.59672367	35.12895245	13.4321173	21.09855025	77.53371144	35.0960567	13.42120978	21.10593359	77.56084401	35.11016506	13.42590647
7	21.06741405	77.355633	35.97233755	13.40140343	21.10876223	77.50745583	31.06575511	13.42770583	21.09657526	77.46270753	35.0420978	13.41995345
8	20.5063012	81.31853695	33.90165268	13.04446832	20.40956588	80.93492926	33.64936315	12.98293305	20.47171989	81.18140342	33.81374304	13.02247046
9	20.41936914	81.00776851	33.68912901	12.9891691	20.50230619	81.33679656	33.91595695	13.04192701	20.50383965	81.34288011	33.91922497	13.04290248
10	20.73178332	80.25178859	33.08324691	13.187902	20.60141716	79.74714712	32.74574305	13.10497348	20.74053084	80.28564985	29.10147102	13.19346648
11	20.72015822	80.26739669	33.09942413	13.18050704	20.7306019	80.30785428	33.12026335	13.18715048	20.72009217	80.26714083	33.09929331	13.18046503
12	21.15817798	78.54147569	31.79331205	13.45914017	21.15817798	78.5414757	31.79331206	13.45914017	21.13590445	78.45879391	35.73839445	13.44497154
13	21.15179917	78.51463419	35.77849544	13.45508249	21.04324398	78.11168167	35.50985621	13.38602836	21.15616665	78.5308461	35.78700249	13.45786073
14	21.14907413	78.53013134	35.7893466	13.45334904	21.15103186	78.53740073	31.79122748	13.45459439	21.13156978	78.46513469	35.75184947	13.44221417
15	21.15524245	78.52943513	35.78646948	13.45727283	21.1534522	78.52278961	35.78297848	13.45613401	21.09979276	78.32360281	31.6457145	13.42200017
16	20.91811128	79.39368307	33.43845518	13.30642895	20.9320127	79.44644526	29.46636611	13.31527192	20.9320127	79.44644526	29.46636611	13.31527192
17	20.95129805	79.33871695	34.38789792	13.32753972	20.93311483	79.26986043	34.35233088	13.31597301	20.94099477	79.29970031	30.35988633	13.32098559
18	21.1665426	78.46263286	35.7399156	13.46446108	21.17304389	78.4867326	31.75295685	13.46859668	21.17304389	78.4867326	31.75295685	13.46859668
19	21.12525242	78.44577386	35.74599128	13.43819557	21.1545999	78.55475176	31.80310469	13.45686409	21.09393469	78.32947969	35.65561525	13.41827374
20	21.02862031	77.32447094	35.95288553	13.37672595	21.10269352	77.59684602	35.14193954	13.4238454	21.09575787	77.5713429	31.11331907	13.4194335
21	20.96885951	77.06062063	31.76676388	13.33871091	20.92701443	76.90683985	32.66295398	13.31209242	21.11039461	77.58076254	35.1214953	13.42874422
22	21.51305698	76.51299375	38.34997196	13.68488581	21.53061568	76.57544274	38.38872412	13.69605525	21.47238505	76.36834062	38.25649956	13.65901358
23	19.82107805	73.29278482	33.61088523	12.55966137	21.51447127	76.50930298	38.34257632	12.55371277	21.50804669	76.48645603	38.32829017	12.56345265
Total	475.7058733	73.29278482	806.6612455	302.5570973	477.2110949	73.39697308	791.9095568	302.431449	477.4832903	73.44074315	796.6137269	302.6184246

A2C

A2C	Time Elapsed (train)				Time Elapsed (test)							
	321.5				22.7							
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	21.50951569	76.63420625	38.45019286	13.68263312	21.50193153	76.60718537	38.43600563	13.67780869	21.49591682	76.58575612	38.4247537	13.67398261
1	21.41560315	76.16763851	38.13840409	13.62289348	21.37462805	76.02190476	38.04907838	13.5968284	21.42034794	76.18451404	38.14875441	13.62591173

2	13.75058829	47.63124821	28.13642285	8.747024226	13.70703677	47.48038822	28.04738554	8.719320228	13.68946405	47.41951734	28.01152894	8.708141872
3	13.79542498	47.80187735	28.26553505	8.775545738	13.83207096	47.92885761	28.34038169	8.798856979	13.81401843	47.86630463	28.30349184	8.787373401
4	13.8429567	47.96654164	28.36256088	8.805781615	13.83357649	47.93403875	28.343373	8.799814679	13.83786969	47.9489149	28.35215344	8.802545668
5	14.17723016	49.16997301	29.15393672	9.018419648	14.17359593	49.15736867	29.14647472	9.016107845	14.12980213	49.00548145	29.05672286	8.988249732
6	21.07393788	77.44326505	35.04916806	13.40555337	21.06242811	77.40096853	35.02727445	13.39823177	21.04504473	77.33708744	34.99422604	13.38717385
7	21.04474662	77.27240238	35.94348169	13.38698422	21.02464636	77.19859796	35.90530953	13.37419804	21.0744324	77.38140307	34.99990387	13.40586794
8	20.4573347	81.12435839	33.80313519	13.01331975	20.45939723	81.13253743	33.8072669	13.01463177	20.42564987	80.99871097	33.73950795	12.9931644
9	20.48656857	81.27436225	33.8840221	13.031916	20.43545571	81.0715872	33.78151706	12.99940209	20.43333746	81.06318365	33.77725992	12.99805462
10	20.69044349	80.09176402	33.00178246	13.16160492	20.68584191	80.07395149	32.99273923	13.15867775	20.68172747	80.05802469	32.98465308	13.15606048
11	20.67680446	80.09944945	33.01403883	13.15292885	20.65907064	80.0307508	32.97915627	13.14164802	20.66586663	80.05707761	32.99252786	13.14597108
12	21.090845	78.2915283	35.6633213	13.41630832	21.07361051	78.22755199	35.63027517	13.40534512	21.09879384	78.32103526	35.67856421	13.42136474
13	21.11203686	78.36703808	35.70198556	13.42978889	21.11464896	78.37673411	35.7069978	13.4314505	21.08676019	78.2732121	35.65350868	13.41370989
14	21.0984095	78.34200488	35.69189067	13.42112025	21.09980681	78.34719332	35.69457105	13.4220091	21.10908648	78.38165034	35.71237545	13.42791209
15	21.1145264	78.37829489	35.70802797	13.43137253	21.10179549	78.33103705	35.68360478	13.42327415	21.10135322	78.32939533	35.68275655	13.42299281
16	20.86490917	79.19175704	33.3350367	13.27258602	20.88040153	79.25055754	33.36510846	13.28244102	20.83443984	79.07611216	33.27586621	13.25320387
17	20.86959073	79.02930633	34.22910607	13.27556406	20.88397874	79.08379109	34.25700221	13.28471656	20.88410546	79.08427094	34.25724785	13.28479717
18	21.09533254	78.19866303	35.60318584	13.41916294	21.10322038	78.22790259	35.61829929	13.42418055	21.12285762	78.30069625	35.6559332	13.43667219
19	21.09958045	78.35044444	35.69670638	13.42186511	21.0801553	78.27831177	35.65944983	13.40950839	21.11846445	78.42056762	35.73294984	13.4338776
20	21.0454662	77.38641508	35.03298269	13.38744196	21.05597523	77.42505788	35.05297055	13.39412696	21.06316238	77.45148579	35.06664274	13.39869885
21	21.07025252	77.43324026	35.04494202	13.40320903	21.06878627	77.42785179	35.04215273	13.40227632	21.08053093	77.47101345	35.06450198	13.40974734
22	21.44708088	76.27834419	38.20540952	13.64291709	21.4003443	76.11212161	38.10351292	13.61318701	21.42759539	76.2090424	38.1629553	13.63052198
23	21.38993603	76.06643342	38.06956829	12.53104655	21.36839369	75.98982501	38.02261288	12.52612977	21.35838352	75.95422704	38.00080356	12.52615791
Total	476.219121	73.24960652	823.1848438	301.8569877	475.9807969	73.21316969	822.69252	301.7141717	475.9990109	73.21577852	821.7295895	301.7321538

DDPG

DDPG	Time Elapsed (Training)	558	Time Elapsed (Testing)	22.1
------	-------------------------	-----	------------------------	------

Step	Episode 0				Episode 1				Episode 2			
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	11.8529288	42.22967191	20.43065746	7.539885071	11.8529288	42.22967191	20.43065746	7.539885071	11.8529288	42.22967191	20.43065746	7.539885071
1	11.86862178	42.2124414	20.42538051	7.549867684	11.86862178	42.2124414	20.42538051	7.549867684	11.86862178	42.2124414	20.42538051	7.549867684
2	12.1609811	42.12494016	22.46218195	7.735843296	12.1609811	42.12494016	22.46218195	7.735843296	12.1609811	42.12494016	22.46218195	7.735843296
3	12.15638712	42.12252445	22.45941092	7.732920974	12.15638712	42.12252445	22.45941092	7.732920974	12.15638712	42.12252445	22.45941092	7.732920974
4	12.15639771	42.12252987	22.45941723	7.732927714	12.15639771	42.12252987	22.45941723	7.732927714	12.15639771	42.12252987	22.45941723	7.732927714
5	12.14347445	42.11642924	22.45202431	7.724706965	12.14347445	42.11642924	22.45202431	7.724706965	12.14347445	42.11642924	22.45202431	7.724706965
6	11.6890395	42.95530287	20.81161505	7.435631808	11.6890395	42.95530287	20.81161505	7.435631808	11.6890395	42.95530287	20.81161505	7.435631808
7	11.66526052	42.83267083	20.73022397	7.42050552	11.66526052	42.83267083	20.73022397	7.42050552	11.66526052	42.83267083	20.73022397	7.42050552
8	11.71916182	46.47279313	23.01420525	7.454793217	11.71916182	46.47279313	23.01420525	7.454793217	11.71916182	46.47279313	23.01420525	7.454793217
9	11.71828482	46.48880666	23.0243976	7.454235341	11.71828482	46.48880666	23.0243976	7.454235341	11.71828482	46.48880666	23.0243976	7.454235341
10	11.77199847	45.5688697	22.44638887	7.488403668	11.77199847	45.5688697	22.44638887	7.488403668	11.77199847	45.5688697	22.44638887	7.488403668
11	11.77026211	45.5965774	22.46356941	7.487299132	11.77026211	45.5965774	22.46356941	7.487299132	11.77026211	45.5965774	22.46356941	7.487299132
12	11.88189608	44.10690058	21.56261679	7.558311737	11.88189608	44.10690058	21.56261679	7.558311737	11.88189608	44.10690058	21.56261679	7.558311737
13	11.8820199	44.10558357	21.56184398	7.558390499	11.8820199	44.10558357	21.56184398	7.558390499	11.8820199	44.10558357	21.56184398	7.558390499
14	11.88101846	44.11625466	21.56810708	7.557753464	11.88101846	44.11625466	21.56810708	7.557753464	11.88101846	44.11625466	21.56810708	7.557753464
15	11.88194077	44.10642418	21.56233721	7.558340164	11.88194077	44.10642418	21.56233721	7.558340164	11.88194077	44.10642418	21.56233721	7.558340164
16	11.81963655	44.86086079	22.01243042	7.518707201	11.81963655	44.86086079	22.01243042	7.518707201	11.81963655	44.86086079	22.01243042	7.518707201
17	11.8255534	44.78119835	21.96426561	7.522471028	11.8255534	44.78119835	21.96426561	7.522471028	11.8255534	44.78119835	21.96426561	7.522471028
18	11.8862262	44.0612631	21.53586706	7.561066211	11.8862262	44.0612631	21.53586706	7.561066211	11.8862262	44.0612631	21.53586706	7.561066211
19	11.88085843	44.11796435	21.56911085	7.557651662	11.88085843	44.11796435	21.56911085	7.557651662	11.88085843	44.11796435	21.56911085	7.557651662
20	11.70745924	43.0495714	20.87412938	7.447348969	11.70745924	43.0495714	20.87412938	7.447348969	11.70745924	43.0495714	20.87412938	7.447348969
21	11.69037772	42.96217264	20.81617225	7.436483072	11.69037772	42.96217264	20.81617225	7.436483072	11.69037772	42.96217264	20.81617225	7.436483072
22	11.86877145	42.21228236	20.42533325	7.549962894	11.86877145	42.21228236	20.42533325	7.549962894	11.86877145	42.21228236	20.42533325	7.549962894
23	21.46785043	76.34351094	38.23975393	7.546979291	11.86981367	42.21117772	20.42500576	7.546979291	11.86981367	42.21117772	20.42500576	7.546979291
Total	294.3464068	45.06948102	536.8714403	181.1304866	284.7483701	43.64730047	519.0566922	181.1304866	284.7483701	43.64730047	519.0566922	181.1304866

SAC

SAC		Time Elapsed (training)			Time (Elapsed (testing))					
		797.4			22.9					
Step	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	
	Water Saved	Water Saved %	Rewards	Water Saved	Water Saved %	Rewards	Water Saved	Water Saved %	Rewards	
0	21.24871898	75.70503848	37.80822606	21.49529635	76.58354551	38.40534293	21.23146851	75.64357841	37.76708253	
1	21.48748676	76.4233028	38.29539801	20.46339253	72.780966	42.84819417	21.52493922	76.5565078	38.37769784	
2	12.50373806	43.31223059	25.28942156	12.74454444	44.14637008	25.80603762	13.02378391	45.11363957	26.4498562	
3	13.88515354	48.11279157	28.44917524	13.89352289	48.14179179	28.46638468	13.89422466	48.14422348	28.46782829	
4	13.81902976	47.88363358	28.31366187	13.92096722	48.23685202	28.52293769	13.58179505	47.06160338	27.83094339	
5	14.22574531	49.33823499	29.25390815	14.24678772	49.41121502	29.29762954	14.24961284	49.4210132	29.30354325	
6	20.32400638	74.6873898	37.1420454	20.43723197	75.10347525	37.42277648	20.39586314	74.95145157	37.33553562	
7	19.12264128	70.21478845	36.94196613	19.11058329	70.17051374	36.93250688	18.79723519	69.01995771	36.13475947	
8	18.04436436	71.55563042	34.26611419	18.05245095	71.58769809	34.28896204	17.98411557	71.31671149	34.1122018	
9	19.08040257	75.69581722	32.34973146	20.32458213	80.63172925	33.49090105	20.02271663	79.43416776	32.72483537	
10	19.68561352	76.20211298	33.01036167	20.48512232	79.29697511	32.56544306	20.42170468	79.05148832	32.32331093	
11	19.89774164	77.08145397	34.91915699	19.64406696	76.09874881	37.23790882	20.00700088	77.50471109	34.18347783	
12	19.97911706	74.16467234	40.84538117	18.29143149	67.89979854	41.64494666	17.7438561	65.86713866	40.24343256	
13	19.00023655	70.52811963	42.61193795	20.0564438	74.44871878	40.09734656	19.54682162	72.5570216	43.75540192	
14	15.91811423	59.10668211	35.69192051	15.33217035	56.93097219	34.21203178	17.59558724	65.33542641	39.89226175	
15	12.73644531	47.27839249	27.71043702	20.09858105	74.60704931	40.13462695	19.77070431	73.38995266	38.28798233	
16	20.11700823	76.35313507	36.36619259	14.98901372	56.8900791	34.16284106	19.2113489	72.91574875	43.08570613	
17	20.02548473	75.83283198	37.00256722	19.42664493	73.56513565	37.50062209	17.79898053	67.40146957	39.19896899	
18	19.25008388	71.35847798	43.13550914	20.15947744	74.72952512	40.22112238	18.69755832	69.31031116	41.56413354	
19	13.44704524	49.93378771	29.51926298	19.03724345	70.69223433	37.18033642	19.26420121	71.53501135	37.72927257	
20	18.39925436	67.65601302	41.44093259	14.04005896	51.62678839	30.65108192	13.54210943	49.79577508	29.42239827	
21	18.48806006	67.94367533	38.42963163	19.60495407	72.04826414	35.21635549	19.91201114	73.17669977	35.85470627	

	22	19.6049688	69.72671789	42.94338754		17.00431532	60.47727541	39.32536386		17.60934303	62.62910727	40.76629959
	23	11.86981367	42.21117772	20.42500576		19.72376578	70.14123436	43.075298		18.85358071	67.04670082	41.03545537
Total		422.1602743	64.92942117	832.1613328		432.5826491	66.51028983	858.7069981		434.6805628	66.84080904	865.8470918

SAC Tuned

SAC Tuned		Time Elapsed (train) 850				Time Elapsed (test) 27							
Step	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 1	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	19.57245292	69.73282968	44.2315793	12.45042875	19.52674883	69.56999491	44.1428678	12.42135546	19.54723617	69.6429873	44.15838359	12.43438787	
1	19.2204172	68.36014747	43.30465607	12.22649179	18.80507525	66.88292473	43.42570084	11.96228447	19.1559167	68.13074227	43.16739874	12.18546173	
2	13.78173385	47.73913462	28.20031694	8.766836536	13.69503142	47.43880237	28.02288565	8.711683384	13.47074659	46.66189262	27.56692198	8.569011322	
3	13.56076722	46.98877581	27.78835526	8.626275245	12.9845178	44.99204108	26.6202019	8.259711464	12.86381939	44.57381468	26.37536888	8.182932787	
4	13.88990856	48.12923221	28.45884686	8.835648636	13.59667303	47.11315632	27.86112517	8.649115646	13.92709603	48.25808865	28.5356345	8.859304326	
5	13.85299978	48.04546571	28.49271775	8.812170221	13.86546888	48.08871149	28.51804318	8.820102066	14.2260638	49.33933958	29.25456702	9.049483703	
6	19.79546814	72.74509845	42.8576819	12.59229319	19.28704669	70.87673299	42.66376229	12.26887614	19.43640216	71.42558987	42.97144594	12.36388414	
7	18.22607998	66.9227818	42.43516365	11.59397399	18.30442409	67.21044685	42.67941432	11.64381025	17.99756921	66.0837327	41.95203434	11.44861373	
8	16.87100403	66.9026243	40.70432313	10.73198309	16.52879285	65.54557251	41.82987303	10.51429571	17.20837849	68.24049586	41.5882663	10.94659372	
9	17.58427061	69.76035906	41.66662725	11.18570622	17.19394923	68.21187519	40.85159103	10.93741499	17.24503846	68.41455647	40.94419361	10.96991386	
10	17.52755315	67.8483596	43.3938696	11.14962711	17.63211013	68.25309493	42.6847663	11.2161379	17.23257687	66.70651988	42.5928964	10.9619868	
11	17.75171722	68.76801389	43.8572364	11.29222236	17.94348322	69.51089226	42.46327294	11.41420854	17.76241646	68.80946145	42.10618733	11.29902836	
12	18.0929948	67.16317979	42.80350106	11.50931585	18.17157041	67.45486106	42.93326367	11.55929937	18.54115905	68.82681459	41.64379051	11.79440209	
13	18.71462828	69.46795312	44.28198495	11.90474934	18.63659193	69.1782853	44.09430187	11.85510886	18.61083535	69.08267793	44.04568674	11.83872458	
14	18.66478778	69.305551	44.17433756	11.8730448	18.56945472	68.95156303	43.9376029	11.81240154	18.79840467	69.80169338	44.41708851	11.95804118	
15	18.79038394	69.75094898	44.38235543	11.95293903	18.68297241	69.35223139	44.16188677	11.88461241	18.60903154	69.0777588	43.98695166	11.83757714	
16	18.51823958	70.28508572	42.8632372	11.77982256	18.51589879	70.27620138	43.85606161	11.77833354	18.35370085	69.6605869	43.54033867	11.67515618	
17	18.20109484	68.92420261	43.98566938	11.57808045	18.14241735	68.7020017	43.80349408	11.54075452	18.1760566	68.82938738	43.8905833	11.56215312	

18	18.53341443	68.70184328	43.71631201	11.78947559	18.38156987	68.13896797	43.35751802	11.69288423	18.33906711	67.98141372	43.27747858	11.66584737
19	18.70666933	69.46469196	44.22847889	11.8996865	18.78935773	69.77174417	44.42378263	11.95228624	18.85196634	70.0042328	43.57459982	11.99211283
20	18.58933825	68.35497166	43.42144636	11.82504985	18.57112436	68.28799725	43.38853229	11.81346363	18.59670682	68.3820667	43.46395144	11.82973715
21	18.79122241	69.05779788	43.8371635	11.9534724	18.59918382	68.35205548	43.39449285	11.83131281	18.07786587	66.43621047	42.11226513	11.49969204
22	18.89728249	67.20977212	43.56906108	12.02093934	19.00628782	67.59745871	43.84824045	12.09027981	18.89007496	67.18413792	43.58453618	12.01635448
23	17.60168847	65.10995675	40.59481275	11.19678607	19.56200296	69.56597689	42.76546025	12.44378132	20.21236518	71.87878111	43.24299922	12.85748974
Total	406.1344288	62.4845342	977.2497543	269.5470189	422.9917536	64.97181625	975.7281418	269.0735143	424.1304947	65.14304096	971.9935684	269.7978903

TQC

TQC		Time Elapsed (train)				Time Elapsed (Test)							
		1312.9				30.4							
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	18.60286239	66.27836784	42.21052517	11.83365282	18.78189294	66.91621875	42.58169797	11.94753774	19.33400661	68.88329199	43.73657547	12.29874828	
1	18.2213914	64.80697011	42.20620888	11.5909915	18.21010834	64.7668403	42.18280724	11.58381412	18.47181939	65.6976529	42.72731456	11.75029375	
2	11.58101063	40.11595578	25.72157504	7.366912483	11.77536341	40.78918265	26.12107521	7.490544175	11.54231188	39.98190551	25.64181859	7.342295432	
3	11.76440611	40.76428951	26.13165597	7.483574013	11.65818982	40.39624446	25.91271296	7.416007708	11.61331525	40.24075172	25.82003992	7.3874621	
4	11.63860817	40.32836305	25.87221375	7.40355143	11.70666954	40.56419914	26.01263852	7.446846631	11.74189562	40.68625926	26.08522591	7.469254641	
5	12.07182553	41.86793394	26.85823323	7.679129658	11.96483293	41.49685838	26.63694305	7.611069521	12.05878821	41.82271743	26.83130081	7.670836357	
6	18.94098278	69.60500491	36.52910795	12.04873797	17.52415719	64.39840325	42.00508824	11.14746687	18.89717543	69.44402007	36.43975358	12.02087123	
7	19.14993709	70.31501358	36.97922733	12.18165798	19.16152207	70.3575515	36.99608166	12.18902742	19.17739538	70.41583534	37.05632275	12.19912475	
8	18.03971253	71.5371834	34.25402662	11.47542193	18.08164149	71.70345435	34.35089433	11.50209379	18.1055466	71.79825097	34.39906322	11.5173003	
9	18.62927694	73.90611059	32.62646135	11.85045565	18.59846051	73.78385556	31.83168161	11.8308527	19.35107151	76.76961568	33.5526519	12.30960361	
10	17.28752225	66.91921095	42.63920829	10.99693866	19.10620508	73.95924932	33.5864392	12.15383918	19.14508628	74.10975665	31.90528173	12.17857228	
11	19.3737789	75.05168542	32.3219403	12.32404823	19.3551677	74.97958788	32.43282705	12.31220928	18.32634382	70.99404812	34.82308854	11.65775383	
12	19.30461359	71.66084152	40.69215366	12.28005079	20.28863743	75.31364587	34.97076477	12.90600804	19.430225	72.12712486	35.25350874	12.35995473	
13	19.02515141	70.6206026	37.21830023	12.10227932	18.75423263	69.61496291	36.60362037	11.92994246	19.22095492	71.34741741	37.57901105	12.22683384	

14	18.7956656	69.79152272	36.7249542	11.9562988	18.64371025	69.22728653	36.33988655	11.85963697	18.62582473	69.16087453	37.2691317	11.84825963
15	18.85149781	69.97780706	36.85840115	11.99181479	19.2895993	71.60406414	37.76605192	12.27049991	18.9994821	70.52713295	37.19407372	12.08595055
16	18.7819204	71.28587356	36.74234253	11.9475552	18.7054148	70.99550024	36.63230362	11.89888846	19.06411755	72.35693924	37.26535237	12.12706646
17	18.89414684	71.5486632	37.80813112	12.01894469	18.7799344	71.11616167	36.6455414	11.94629187	19.40287705	73.47513107	37.90615517	12.34255815
18	18.95348432	70.2590078	37.02741772	12.05669045	19.40406293	71.92926565	37.88643239	12.34331251	19.36950685	71.80116912	37.85490984	12.3213307
19	18.92309653	70.26836515	37.00634764	12.03736017	18.90913061	70.21650458	37.00952278	12.02847616	17.76902535	65.98287758	42.13351683	11.30323241
20	18.84765773	69.30483983	36.32905859	11.98937203	19.38953497	71.29738001	37.49510051	12.33407098	19.03938786	70.00985189	36.81598667	12.1113354
21	19.11861135	70.26095318	36.99540207	12.16173105	19.10870498	70.22454726	36.96305832	12.15542941	19.13159819	70.30867985	37.01655303	12.16999224
22	17.07087495	60.71400031	39.46691497	10.85912497	18.19877379	64.72546725	42.15721545	11.57660398	17.17569328	61.08679549	39.65943036	10.92580201
23	17.39275591	61.85174688	40.0550688	11.12623498	17.05336132	60.6447991	39.43274804	11.19563027	17.02826874	60.5555654	39.36096491	11.21357412
Total	419.2607911	64.54334637	857.2748766	266.7625296	422.4493084	65.04255128	850.5531332	269.0761001	422.0217176	64.98265271	854.3270314	268.8380068

TQC Tuned

Step	Time Elapsed (training)				Time (Elapsed) (testing)							
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	%	Rewards	Carbon Emissions	Water Saved	%	Rewards	Carbon Emissions
0	19.71779429	70.25065261	43.54888221	12.5428833	19.27131453	68.65993236	43.54776824	12.2588686	19.26067951	68.62204185	43.58234591	12.25210345
1	18.86940552	67.11172452	43.5611374	12.00320624	19.12187641	68.00967314	43.09508305	12.16380802	19.04234589	67.72681154	42.9264422	12.11321707
2	13.80492077	47.81945278	28.24809101	8.7815862	13.77045785	47.7000752	28.17715466	8.759663647	13.62317002	47.18987862	27.87650171	8.665970913
3	12.93356163	44.81547524	26.51684563	8.227297223	12.90442078	44.71450068	26.45775817	8.208760149	13.50612669	46.79944349	27.67749943	8.591517313
4	13.91394464	48.21251842	28.50840638	8.850938464	13.78465965	47.76453943	28.24348692	8.768697697	13.88386574	48.10829353	28.4464248	8.831804676
5	13.90042284	48.20994006	28.58911788	8.84233698	12.93876078	44.87466954	28.63468035	8.230604506	14.00391747	48.56888379	28.7997915	8.908171979
6	19.65887189	72.24312964	43.52017231	12.50540159	19.88697681	73.08137779	43.09572141	12.65050369	19.90192439	73.13630768	43.12463465	12.66001214

7	18.21909629	66.89713899	42.46122737	11.58953153	18.16936291	66.71452726	42.36162197	11.55789514	18.02861342	66.19772126	42.92622238	11.46836157
8	16.71843486	66.29760528	41.33140483	10.63493078	16.80076879	66.62410371	42.54180563	10.68730504	16.59341959	65.80185236	41.99957642	10.55540607
9	17.39327434	69.00263821	39.40067464	11.06420967	17.74793367	70.40964352	41.02866458	11.28981556	17.40181677	69.03652778	40.34239002	11.06964369
10	17.68507982	68.4581382	43.67877298	11.24983298	17.74903356	68.7057002	43.82747575	11.29051523	17.48063679	67.66674854	43.3616954	11.11978268
11	17.79930811	68.95237528	42.13799769	11.32249588	18.14685516	70.29873067	41.85024088	11.54357751	17.6779486	68.48224316	43.7779585	11.24529667
12	17.74555051	65.87342849	41.93413622	11.28829959	18.2879694	67.88694686	43.21635668	11.63334309	17.97274714	66.71680728	42.376049	11.43282391
13	18.70404944	69.4286849	44.25466569	11.89801993	18.7052386	69.43309902	44.23544073	11.89877638	18.68124404	69.34403218	44.15535602	11.88351296
14	18.68449587	69.37873049	44.23210787	11.88558151	18.71715182	69.49998763	44.29965823	11.90635461	18.65525847	69.27016704	44.08964978	11.86698302
15	18.66741803	69.2944927	44.15895801	11.87471796	18.57083204	68.93596012	43.89804724	11.81327768	18.65871589	69.26218988	44.14162227	11.86918236
16	18.43538206	69.97060399	43.65796839	11.72711523	18.55926464	70.44079436	43.91114893	11.80591942	18.50821675	70.24704457	42.84927061	11.77344684
17	18.12699523	68.64360098	43.81567458	11.53094421	18.28269779	69.23321799	44.17199738	11.62998972	18.22416569	69.01156765	43.91767048	11.59275628
18	18.58629359	68.89786197	43.86786979	11.82311308	18.49686247	68.56634815	43.62854919	11.76622415	18.49389033	68.55533066	43.62546418	11.76433352
19	18.64935871	69.25187671	44.13672572	11.86323006	18.69294344	69.4137227	44.16929773	11.89095518	18.7587978	69.658264	44.3489425	11.93284646
20	18.64364892	68.55467775	43.59591201	11.85959795	18.39404602	67.63686139	43.05066958	11.70082055	18.74677801	68.9338944	43.79689777	11.92520043
21	18.34265224	67.40930119	42.81164301	11.66812795	18.66754625	68.60328762	43.58699698	11.87479952	18.9207716	69.53389153	42.11188982	12.03588123
22	18.79238254	66.83668663	43.38567564	11.95421038	18.93388792	67.33996239	43.69522716	12.04422478	18.8619888	67.08424714	43.43756946	11.99848832
23	20.11105951	71.51852005	43.02153052	12.79304717	20.08371637	71.42128296	42.95436775	12.77565366	20.18232324	71.77194661	43.17343131	12.83837946
Total	424.1034017	65.13871896	974.3755978	269.7806559	424.6845777	65.24870603	977.6792192	270.1503535	425.0693627	65.28025569	976.8652961	270.395123

ARS

ARS	Time Elapsed (train) 301.1				Time Elapsed (test) 24.6							
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	17.1002379	60.92481008	39.55981021	10.87780333	17.1002379	60.92481008	39.55981021	10.87780333	17.1002379	60.92481008	39.55981021	10.87780333
1	17.11039046	60.85553727	39.50916588	10.88426158	17.11039046	60.85553727	39.50916588	10.88426158	17.11039046	60.85553727	39.50916588	10.88426158
2	12.18072005	42.19331477	26.94976708	7.74839964	12.18072005	42.19331477	26.94976708	7.74839964	12.18072005	42.19331477	26.94976708	7.74839964

3	12.30590698	42.64061867	27.2407878	7.828033546	12.30590698	42.64061867	27.2407878	7.828033546	12.30590698	42.64061867	27.2407878	7.828033546
4	12.27729399	42.54144157	27.18233588	7.809832256	12.27729399	42.54144157	27.18233588	7.809832256	12.27729399	42.54144157	27.18233588	7.809832256
5	12.57087595	43.59875833	27.88374927	7.996585612	12.57087595	43.59875833	27.88374927	7.996585612	12.57087595	43.59875833	27.88374927	7.996585612
6	19.44912143	71.47233108	37.58562274	12.37197512	19.44912143	71.47233108	37.58562274	12.37197512	19.44912143	71.47233108	37.58562274	12.37197512
7	19.42190932	71.31364512	37.48835722	12.35466495	19.42190932	71.31364512	37.48835722	12.35466495	19.42190932	71.31364512	37.48835722	12.35466495
8	18.3700879	72.84729983	34.8665005	11.68558032	18.3700879	72.84729983	34.8665005	11.68558032	18.3700879	72.84729983	34.8665005	11.68558032
9	18.32652881	72.70504752	34.79280054	11.65787151	18.32652881	72.70504752	34.79280054	11.65787151	18.32652881	72.70504752	34.79280054	11.65787151
10	18.6496886	72.19209483	37.32588856	11.86343991	18.6496886	72.19209483	37.32588856	11.86343991	18.6496886	72.19209483	37.32588856	11.86343991
11	18.64950795	72.24594702	37.36162262	11.863325	18.64950795	72.24594702	37.36162262	11.863325	18.64950795	72.24594702	37.36162262	11.863325
12	19.28820941	71.59994742	37.74625409	12.26961577	19.28820941	71.59994742	37.74625409	12.26961577	19.28820941	71.59994742	37.74625409	12.26961577
13	19.31048391	71.67974545	37.79160837	12.28378503	19.31048391	71.67974545	37.79160837	12.28378503	19.31048391	71.67974545	37.79160837	12.28378503
14	19.30503403	71.68289488	37.79494493	12.28031825	19.30503403	71.68289488	37.79494493	12.28031825	19.30503403	71.68289488	37.79494493	12.28031825
15	19.30941852	71.67763416	37.79065709	12.28310731	19.30941852	71.67763416	37.79065709	12.28310731	19.30941852	71.67763416	37.79065709	12.28310731
16	18.96027756	71.96281955	37.07134574	12.06101176	18.96027756	71.96281955	37.07134574	12.06101176	18.96027756	71.96281955	37.07134574	12.06101176
17	18.98111555	71.87799771	37.01048502	12.07426722	18.98111555	71.87799771	37.01048502	12.07426722	18.98111555	71.87799771	37.01048502	12.07426722
18	19.32092153	71.62106731	37.75235366	12.2904246	19.32092153	71.62106731	37.75235366	12.2904246	19.32092153	71.62106731	37.75235366	12.2904246
19	19.304877	71.68605539	37.79699357	12.28021836	19.304877	71.68605539	37.79699357	12.28021836	19.304877	71.68605539	37.79699357	12.28021836
20	19.40127741	71.34055822	37.51903853	12.34154059	19.40127741	71.34055822	37.51903853	12.34154059	19.40127741	71.34055822	37.51903853	12.34154059
21	19.41485893	71.34966393	37.51663871	12.35018006	19.41485893	71.34966393	37.51663871	12.35018006	19.41485893	71.34966393	37.51663871	12.35018006
22	17.38576029	61.83391647	40.04946407	11.05942984	17.38576029	61.83391647	40.04946407	11.05942984	17.38576029	61.83391647	40.04946407	11.05942984
23	18.62488247	66.23340904	40.4756243	11.24801099	17.39275591	61.85174688	40.0550688	11.24801099	17.39275591	61.85174688	40.0550688	11.24801099
Total	425.019386	65.41985648	864.0618164	269.7636825	423.7872594	65.23728723	863.6412609	269.7636825	423.7872594	65.23728723	863.6412609	269.7636825

ARS Tuned

ARS Tuned				Time Elapsed (train)				Time Elapsed (Test)			
				335				26.6			
Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2

Step	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	18.8185959	67.04698423	41.1595925	11.97088522	18.8185959	67.04698423	41.1595925	11.97088522	18.8185959	67.04698423	41.1595925	11.97088522
1	18.84650861	67.03028837	41.14727993	11.98864106	18.84650861	67.03028837	41.14727993	11.98864106	18.84650861	67.03028837	41.14727993	11.98864106
2	13.39516417	46.40007946	27.41366435	8.520931833	13.39516417	46.40007946	27.41366435	8.520931833	13.39516417	46.40007946	27.41366435	8.520931833
3	13.49392083	46.75714951	27.65272221	8.583752917	13.49392083	46.75714951	27.65272221	8.583752917	13.49392083	46.75714951	27.65272221	8.583752917
4	13.49532632	46.76198489	27.65548281	8.584646981	13.49532632	46.76198489	27.65548281	8.584646981	13.49532632	46.76198489	27.65548281	8.584646981
5	13.8051312	47.8794462	28.39541092	8.781720059	13.8051312	47.8794462	28.39541092	8.781720059	13.8051312	47.8794462	28.39541092	8.781720059
6	18.4167721	67.67861663	41.59848111	11.71527707	18.4167721	67.67861663	41.59848111	11.71527707	18.4167721	67.67861663	41.59848111	11.71527707
7	18.49531048	67.91134624	41.74482678	11.76523691	18.49531048	67.91134624	41.74482678	11.76523691	18.49531048	67.91134624	41.74482678	11.76523691
8	18.08330217	71.71003982	40.49335725	11.50315018	18.08330217	71.71003982	40.49335725	11.50315018	18.08330217	71.71003982	40.49335725	11.50315018
9	18.26293172	72.452745	38.98710995	11.61741613	18.26293172	72.452745	38.98710995	11.61741613	18.26293172	72.452745	38.98710995	11.61741613
10	18.44572601	71.40256494	43.19748972	11.73369523	18.44572601	71.40256494	43.19748972	11.73369523	18.44572601	71.40256494	43.19748972	11.73369523
11	18.39519134	71.26075513	43.10681988	11.70154912	18.39519134	71.26075513	43.10681988	11.70154912	18.39519134	71.26075513	43.10681988	11.70154912
12	18.73699618	69.55378349	41.86259769	11.91897801	18.73699618	69.55378349	41.86259769	11.91897801	18.73699618	69.55378349	41.86259769	11.91897801
13	18.65541106	69.24814117	41.66281851	11.86708008	18.65541106	69.24814117	41.66281851	11.86708008	18.65541106	69.24814117	41.66281851	11.86708008
14	18.65257313	69.2601959	41.67146887	11.86527482	18.65257313	69.2601959	41.67146887	11.86527482	18.65257313	69.2601959	41.67146887	11.86527482
15	18.65572295	69.25107988	41.66488909	11.86727848	18.65572295	69.25107988	41.66488909	11.86727848	18.65572295	69.25107988	41.66488909	11.86727848
16	18.47481669	70.12027624	42.29278588	11.75220039	18.47481669	70.12027624	42.29278588	11.75220039	18.47481669	70.12027624	42.29278588	11.75220039
17	18.53522542	70.18949371	42.33145723	11.79062759	18.53522542	70.18949371	42.33145723	11.79062759	18.53522542	70.18949371	42.33145723	11.79062759
18	18.70559778	69.34011278	41.7197977	11.89900486	18.70559778	69.34011278	41.7197977	11.89900486	18.70559778	69.34011278	41.7197977	11.89900486
19	18.64944397	69.25219331	41.66640927	11.8632843	18.64944397	69.25219331	41.66640927	11.8632843	18.64944397	69.25219331	41.66640927	11.8632843
20	18.58137971	68.32570727	42.02450245	11.81998726	18.58137971	68.32570727	42.02450245	11.81998726	18.58137971	68.32570727	42.02450245	11.81998726
21	18.51211416	68.03207421	41.82797267	11.77592606	18.51211416	68.03207421	41.82797267	11.77592606	18.51211416	68.03207421	41.82797267	11.77592606
22	18.8965879	67.20730175	41.24312855	12.0204975	18.8965879	67.20730175	41.24312855	12.0204975	18.8965879	67.20730175	41.24312855	12.0204975
23	18.67144668	66.39899969	40.7494113	11.87728066	18.67144668	66.39899969	40.7494113	11.87728066	18.67144668	66.39899969	40.7494113	11.87728066
Total	425.6811965	65.43630666	943.2694766	270.7843227	425.8531109	65.46177992	943.6355401	270.8936809	425.8531109	65.46177992	943.6355401	270.8936809

Appendix G: Background Leakage – SZ08 Results

This section shows the episodic performance of each of the optimisation algorithms as they tackle the background leakage case study on the Jowitt & Xu network. The results and discussions associated with these results were covered in section 4.2 of the thesis.

Each algorithm’s step rewards, water saved% and carbon emissions are listed along with the algorithm’s processing speeds and episodic pressure violations. A line plot of the rewards of the algorithms across the three test episodes is shown below.



NM

NM		Time Elapse 23238				Violations 2284							
Step	Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	9.531886445	0.552464962	85.00172388	6.063423605	10.58157957	0.613304825	84.63056521	6.731154396	10.5545117	0.611735981	85.86378358	6.713935985	
1	9.931611278	0.573607447	90.70500913	6.317696566	9.929568062	0.57348944	90.83150303	6.316396835	9.763674172	0.563908117	90.04448112	6.210868414	
2	10.5865765	0.614268784	90.43132805	6.734333041	9.934849933	0.576453417	90.47102178	6.319756739	10.58739632	0.614316353	91.90486468	6.734854547	
3	9.555782525	0.572987707	86.8689942	6.07862438	8.054148673	0.482946129	74.35706532	5.123405054	10.5924983	0.635151679	93.14635252	6.738100018	
4	2.870905078	0.180614385	82.17039998	1.826240138	2.045282464	0.128672814	83.22475823	1.301045081	2.591107732	0.163011774	85.04729327	1.648255451	
5	10.32249919	0.638035674	91.31630554	6.566348185	10.59901838	0.655127379	89.95341359	6.742247574	10.59287845	0.654747868	90.09990103	6.73834184	
6	10.31360454	0.618556531	91.43270761	6.560690122	10.58752914	0.634985108	90.08821637	6.734939037	9.928590043	0.595465357	90.08245418	6.315774698	
7	16.7224889	1.004207328	93.26850296	10.63750964	16.45187339	0.987956513	93.08262912	10.4653657	16.44660114	0.987639907	91.67475566	10.46201191	
8	9.924345961	0.612819074	88.69688872	6.313074953	9.537483601	0.588930685	85.28006345	6.066984068	10.29817652	0.635902761	89.83087663	6.550876049	
9	9.929407523	0.63127806	87.64214166	6.316294713	9.761167844	0.620581952	88.80712407	6.209274089	9.92917185	0.631263077	87.45075395	6.316144797	
10	10.67140902	0.708765484	96.10456727	6.788296709	8.723971143	0.579422044	85.98491172	5.549492523	10.66690242	0.708466168	95.71326671	6.78542997	
11	10.54282522	0.686767835	86.96031079	6.706501978	9.882604095	0.643760517	88.60780924	6.286522117	10.25047158	0.667723691	88.88874814	6.520529979	
12	12.11081792	0.776089305	84.88581958	7.703933493	12.1107221	0.776083166	84.89742523	7.703872545	11.82050714	0.757485517	87.12594648	7.519261001	
13	12.08323596	0.770218389	85.52316601	7.686388057	11.43587427	0.72895379	84.80435189	7.27458834	11.43919937	0.729165742	86.25095566	7.276703506	
14	11.06691933	0.688417012	82.42696923	7.039888723	11.45827683	0.712761381	86.44400914	7.288839057	11.28682257	0.702096079	83.47229889	7.179773571	
15	9.753383477	0.578250629	86.98474329	6.204322298	10.57801576	0.627140754	87.78134067	6.728887382	10.57045792	0.626692671	91.24581092	6.72407969	
16	10.57652481	0.691052919	89.44944365	6.72793896	9.746556958	0.636824171	87.11466516	6.199979812	10.08699139	0.659067603	87.38654142	6.416536965	
17	9.003256962	0.598189287	81.10508228	5.727151818	10.56829284	0.702172513	89.23252804	6.722702444	10.43533274	0.693338453	89.61620657	6.638123864	
18	8.625658357	0.564334947	72.97380216	5.486953794	10.33692555	0.676294851	86.99021323	6.575525083	10.62203968	0.694948484	84.04403432	6.75689188	
19	9.91991831	0.654885735	88.58228796	6.310258435	9.919132959	0.654833889	88.50975134	6.309758858	10.57403479	0.698068707	86.2541148	6.726355009	
20	9.88011125	0.644408846	86.54947156	6.284936369	9.879278917	0.644354559	86.44826349	6.284406905	9.880282605	0.644420023	86.53463236	6.285045371	
21	10.68640163	0.655764356	87.77175009	6.797833804	9.751616135	0.598401828	86.63068396	6.203198056	9.919753759	0.608719488	88.21528015	6.310153761	

22	10.41670311	0.638003139	86.79952551	6.626273182	10.69642917	0.655135825	86.89188275	6.804212523	10.69591687	0.655104448	86.80703959	6.803886641
23	10.21853362	0.637129906	87.11525831	6.494483626	2.956565791	0.191915019	81.57464106	6.415379168	2.307353094	0.149773671	81.60035943	6.616729467
Total	245.2448069	0.637129906	2090.766199	155.9993966	235.5267636	0.612104274	2082.638837	154.3579334	241.8406722	0.628675567	2118.300752	158.9886644

PSO

PSO	Time Elapsed				Violations							
	12909.4				22849							
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	10.56567384	0.612382935	88.54451673	6.721036446	10.55442048	0.611730693	88.47796487	6.713877953	10.55349878	0.611677272	88.28937777	6.713291646
1	9.738561204	0.5624577	90.81058742	6.194893553	10.28591862	0.594070726	93.605678	6.54307855	10.576249	0.610838969	93.73391108	6.727763515
2	10.30821938	0.598117568	92.68209382	6.55726451	10.56823523	0.613204562	92.37597631	6.722665793	10.58662694	0.614271711	93.72665848	6.734365127
3	10.31952919	0.618783794	90.94557404	6.564458905	10.28734124	0.616853728	92.41309788	6.543983511	9.685724411	0.580779335	89.93233783	6.161283012
4	2.860538541	0.179962205	85.49782891	1.819645776	2.842029607	0.178797771	84.29653606	1.807871874	2.84678278	0.179096803	85.0422575	1.810895462
5	10.32010131	0.637887461	93.0100374	6.564822845	10.57350537	0.653550415	92.67858735	6.726018237	10.5416726	0.651582825	92.0728988	6.705768776
6	10.5620942	0.633459652	91.76126326	6.718759364	10.57478749	0.63422093	93.14411203	6.72683382	10.31129535	0.618418038	92.46613623	6.5592212
7	16.72083019	1.004107721	93.56342775	10.6364545	16.45071544	0.987886977	93.07061408	10.46462911	16.41342969	0.985647919	92.09309314	10.4409109
8	10.5494279	0.651417298	89.9107144	6.710702074	10.55456437	0.65173447	88.40203429	6.713969486	10.29731285	0.63584943	89.84255831	6.550326651
9	10.58623724	0.673037066	89.22268538	6.734117231	10.57204828	0.67213498	90.96804157	6.725091353	10.57281691	0.672183847	90.99342812	6.725580291
10	10.49067735	0.696761786	97.12206138	6.673329676	10.13738047	0.673296784	95.6173204	6.448590465	10.55648699	0.70113268	97.55825815	6.715192502
11	10.44131525	0.680155397	88.53388987	6.641929458	9.885104699	0.643923409	88.91432349	6.288112801	9.868379165	0.642833894	88.47392112	6.277473354
12	11.44134464	0.733187905	86.31907732	7.278068153	11.45284682	0.733924991	86.71692544	7.285384917	11.43447875	0.732747923	86.09866152	7.273700625
13	12.07561664	0.769732712	86.44815065	7.681541256	12.06868727	0.769291016	87.03867477	7.677133349	11.80482438	0.752471676	86.68994908	7.509284885
14	11.45801939	0.712745367	86.34313028	7.288675296	11.79544444	0.733734871	86.60128107	7.503318117	11.82949819	0.735853182	86.50643751	7.524980388
15	10.55047693	0.625508054	89.44883455	6.7111369383	10.56568492	0.626409694	91.18884933	6.721043494	9.525189069	0.564721626	87.62944181	6.05916327
16	10.28945332	0.672296135	88.99872499	6.545327043	10.27667068	0.671460939	89.7454731	6.537195753	10.28500203	0.672005295	90.23238992	6.542495488
17	9.694984585	0.644148661	87.94077275	6.167173594	10.56272456	0.701802548	87.43243557	6.719160345	9.691015483	0.643884948	87.20665881	6.164648769

18	9.955766006	0.651357433	86.51442638	6.333061871	10.62164148	0.694922432	85.84168052	6.75663858	10.3251439	0.675524034	85.99069217	6.568030535
19	10.53447304	0.695456949	87.74825966	6.701188987	10.55793768	0.69700602	89.1537182	6.716115315	10.26931434	0.677951901	88.83656211	6.53251624
20	10.68155443	0.696681241	88.26612991	6.794750404	10.65538842	0.69497462	87.50971014	6.778105681	10.42507746	0.679953086	88.39588936	6.631600271
21	10.28238004	0.630971824	88.54832676	6.540827593	10.28947905	0.63140745	88.76054237	6.545343415	10.25634844	0.62937441	88.28785185	6.524268368
22	10.65373391	0.652520822	88.57507675	6.777053214	9.868819563	0.60444632	86.45440653	6.2777535	10.65301248	0.652476636	88.20799509	6.776594298
23	2.957483415	0.191974583	80.12912712	6.662699906	2.662306637	0.172814226	82.73729334	6.71200736	2.657749902	0.172518441	81.11326466	6.611489149
Total	244.0384919	0.634379678	2136.884717	160.019151	244.6636828	0.635983357	2143.145277	160.6539228	241.9669299	0.628908162	2139.42063	158.8408447

DE

DE	Time Elapsed				7013.1				Violations				22867			
	Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2			
Step	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	10.56053153	0.612084888	88.52700911	6.717765315	10.56053153	0.612084888	88.52700911	6.717765315	10.52198947	0.609851003	88.19855954	6.693247942				
1	10.55365218	0.609533873	93.96673547	6.713389225	10.55365218	0.609533873	93.96673547	6.713389225	10.48797594	0.605740694	93.41755876	6.671611254				
2	10.30473425	0.597915349	93.20957856	6.555047551	10.30473425	0.597915349	93.20957856	6.555047551	10.59453685	0.614730671	92.09372897	6.73939678				
3	10.27693973	0.616230028	92.25834632	6.5373669	10.27693973	0.616230028	92.25834632	6.5373669	10.57209786	0.633928419	92.94277369	6.725122889				
4	2.87161608	0.180659115	83.92062635	1.826692421	2.87161608	0.180659115	83.92062635	1.826692421	2.871912359	0.180677755	84.01578518	1.82688089				
5	10.31640176	0.637658791	93.07007355	6.562469487	10.31640176	0.637658791	93.07007355	6.562469487	10.58831497	0.654465799	93.40448123	6.735438918				
6	10.29902931	0.617682384	92.91429523	6.551418527	10.29902931	0.617682384	92.91429523	6.551418527	10.53209344	0.631660362	92.85175452	6.699675277				
7	16.47167814	0.989145814	92.40623922	10.4779639	16.47167814	0.989145814	92.40623922	10.4779639	16.45933772	0.988404756	92.02582945	10.47011391				
8	10.50977563	0.648968808	89.3584149	6.685478472	10.50977563	0.648968808	89.3584149	6.685478472	10.58131325	0.653386189	88.76268844	6.730984987				
9	10.58605707	0.673025612	89.36674068	6.734002625	10.58605707	0.673025612	89.36674068	6.734002625	10.56269703	0.67154046	90.90685647	6.719142837				
10	10.65236391	0.70750056	97.23859693	6.77618173	10.65236391	0.70750056	97.23859693	6.77618173	10.63101783	0.706082813	96.28910971	6.762603063				
11	10.15019115	0.661191346	88.28535427	6.456739596	10.15019115	0.661191346	88.28535427	6.456739596	9.858363244	0.642181449	88.81316615	6.271102027				
12	12.087498	0.774594911	86.34863408	7.689099229	12.087498	0.774594911	86.34863408	7.689099229	11.43018254	0.732472611	86.60213835	7.270967715				
13	12.0714258	0.769465577	87.10307269	7.67887538	12.0714258	0.769465577	87.10307269	7.67887538	12.07692453	0.769816081	86.19880226	7.682373233				

14	12.09720251	0.752505712	87.15386039	7.695272463	12.09720251	0.752505712	87.15386039	7.695272463	12.10077356	0.752727849	87.26162843	7.697544079
15	10.57800636	0.627140197	89.48317774	6.728881408	10.57800636	0.627140197	89.48317774	6.728881408	10.15849546	0.602268578	89.51912942	6.462022134
16	10.22270232	0.667934733	90.04773447	6.502865397	10.22270232	0.667934733	90.04773447	6.502865397	10.22003511	0.667760463	89.68834808	6.501168737
17	10.56780701	0.702140233	87.65054304	6.722393398	10.56780701	0.702140233	87.65054304	6.722393398	10.29625368	0.684097841	88.77379568	6.549652892
18	10.58894368	0.692783174	86.97760806	6.735838856	10.58894368	0.692783174	86.97760806	6.735838856	10.58491036	0.692519293	87.23514244	6.733273178
19	10.29566415	0.679691442	89.11060081	6.549277877	10.29566415	0.679691442	89.11060081	6.549277877	10.57789506	0.698323551	87.81089225	6.728810604
20	10.65088305	0.694680767	88.72464116	6.775239725	10.65088305	0.694680767	88.72464116	6.775239725	10.65376913	0.694869005	88.68049911	6.777075621
21	10.58441321	0.6495059	88.33012395	6.732956933	10.58441321	0.6495059	88.33012395	6.732956933	10.68613933	0.65574826	89.21354487	6.797666953
22	10.66488942	0.653204076	88.50780048	6.78414946	10.66488942	0.653204076	88.50780048	6.78414946	10.66098555	0.652964971	88.55979306	6.781666129
23	10.60706114	0.661532317	89.73738293	6.747363734	10.60443258	0.661910873	89.73738293	6.745691655	2.952273125	0.191636375	89.70721766	1.87799998
Total	243.9624063	0.661532317	2153.69719	161.9367296	243.9624063	0.63396847	2153.69719	161.9350575	246.6602874	0.641160635	2152.973224	156.905542

TRPO

TRPO	Time Elapsed (training) 2666.6				Time (Elapsed (testing) 835.7				Violations 2295			
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	10.57449707	0.612894326	83.21515364	6.726649077	10.57386796	0.612857863	83.00674357	6.726248886	10.57146782	0.612718752	79.82686431	6.724722111
1	10.57872196	0.610981796	90.90009698	6.729336613	10.57848161	0.610967915	90.89529474	6.729183723	10.57846138	0.610966746	90.89555769	6.729170851
2	10.58393894	0.614115744	90.50208777	6.732655239	10.58583868	0.614225974	90.53286848	6.7338637	10.58393894	0.614115744	90.50208777	6.732655239
3	10.59360963	0.635218317	89.65259694	6.738806957	10.59201408	0.635122644	89.63955133	6.737791996	10.59102737	0.635063478	89.6155293	6.737164328
4	2.861248583	0.180006875	79.20039952	1.820097448	2.861861937	0.180045463	82.20330373	1.820487616	2.862142038	0.180063084	82.16903293	1.820665793
5	10.58891366	0.654502804	87.16954289	6.735819755	10.58893736	0.654504269	87.18154232	6.735834836	10.58893736	0.654504269	87.18154232	6.735834836
6	10.58504333	0.634836022	90.08626119	6.73335776	10.58495514	0.634830733	90.09035695	6.733301661	10.58872092	0.635056585	90.12465017	6.735697151
7	16.72478283	1.004345082	90.30641515	10.63896885	16.72498938	1.004357486	90.31809823	10.63910025	16.72346882	1.004266174	90.29599768	10.63813299
8	24.30037023	1.500525116	62.97244051	15.45795151	10.57436202	0.652956957	87.16573802	6.726563167	24.30105754	1.500567557	65.98188307	15.45838872
9	10.57967395	0.672619795	87.73059321	6.729942194	10.57804627	0.672516312	84.69674862	6.728906793	10.58118202	0.672715672	87.74913065	6.730901504

10	10.58850284	0.703259084	92.54013097	6.735558427	10.62282895	0.70553893	69.21381196	6.757393955	10.62998869	0.70601446	90.06755782	6.761948403
11	10.53418141	0.68620477	82.46522523	6.701003479	10.53445153	0.686222366	85.46919044	6.701175305	10.53418141	0.68620477	82.46522523	6.701003479
12	12.10146906	0.775490209	83.30446766	7.697986501	12.10146859	0.775490179	83.30405765	7.697986201	12.10151939	0.775493435	83.29101693	7.698018517
13	12.08861933	0.770561539	82.74134517	7.689812526	12.09115727	0.770723315	82.78104848	7.691426965	12.09136334	0.77073645	82.79285592	7.691558051
14	12.10953101	0.753272606	84.0451987	7.703114867	12.10722822	0.753129361	84.01696542	7.701650014	12.1093317	0.753260208	84.04978986	7.70298808
15	10.56888778	0.626599582	87.9219604	6.723080894	10.56880905	0.626594914	87.89893573	6.723030814	10.56888778	0.626599582	87.9219604	6.723080894
16	10.56746546	0.690460996	87.86459071	6.722176126	10.56758377	0.690468727	87.87438709	6.722251389	10.65697504	0.696309407	85.30816763	6.779114964
17	10.41941957	0.692281159	17.10377978	6.628001178	10.56172342	0.70173603	88.24399558	6.7185235	10.50442215	0.697928852	59.40302992	6.68207302
18	10.57213595	0.691683526	83.12010468	6.725147123	10.57214071	0.691683837	83.09009774	6.725150148	10.5720983	0.691681062	83.00615783	6.725123173
19	10.57428648	0.698085323	86.14942708	6.726515118	10.57189141	0.697927207	86.09427562	6.724991561	10.57189141	0.697927207	86.09427562	6.724991561
20	10.70497117	0.698208547	85.84587715	6.809646261	10.70521455	0.698224421	83.87649992	6.809801079	10.7056408	0.698252223	85.89250058	6.810072227
21	10.69157868	0.656082042	84.01003438	6.801127028	10.69152874	0.656078978	86.02909681	6.801095264	10.69261969	0.656145923	86.04419524	6.80178924
22	10.69189864	0.654858339	85.49292675	6.801330566	10.69320808	0.654938539	85.52066677	6.802163522	10.69270648	0.654907817	85.4989427	6.801844448
23	11.31233685	0.705091026	82.36263724	7.196003718	10.72315603	0.668745323	80.11208433	6.821214015	11.32182741	0.705717368	80.10993277	7.202040851
Total	271.4960844	0.705091026	1976.703294	172.7040892	257.3557448	0.668745323	2049.25536	163.7091364	271.7238578	0.705717368	1649.983022	172.8489804

PPO

PPO		Time Elapsed (train) 3652				Time Elapsed (Test) 741.9				Violations 12604			
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	10.57392216	0.612861005	76.17881734	6.726283365	10.57406752	0.61286943	76.16287025	6.726375834	10.57496202	0.612921274	76.29873398	6.726944838	
1	10.57964373	0.611035034	80.92408371	6.72992297	10.578659	0.61097816	80.89714484	6.729296563	10.57874365	0.610983049	80.9245814	6.729350408	
2	10.58509547	0.61418285	80.51266008	6.733390928	10.58406122	0.61412284	80.49677351	6.732733026	10.58431157	0.614137366	80.53298122	6.732892278	
3	10.59125282	0.635076997	79.61312701	6.737307746	10.59097231	0.635060177	79.61453588	6.737129307	10.59185728	0.635113242	79.63865464	6.737692252	
4	2.864019207	0.180181181	59.1895653	1.821859898	2.861185099	0.180002881	57.20280302	1.820057065	2.862843327	0.180107204	59.22385539	1.821111897	
5	10.58893736	0.654504269	78.18154232	6.735834836	10.58984847	0.654560585	80.18895781	6.736414406	10.58878825	0.654495052	78.16129513	6.735739982	

6	10.5867856	0.634940514	80.11945978	6.734466054	10.58495559	0.63483076	80.09075228	6.73330195	10.58840381	0.635037566	80.10340889	6.735495431
7	16.72411201	1.004304798	80.31943821	10.63854213	16.72341634	1.004263022	80.31119766	10.6380996	16.72288383	1.004231044	80.29578271	10.63776086
8	24.30040819	1.50052746	69.97153679	15.45797566	24.30180564	1.500613751	71.98451696	15.45886461	24.30225281	1.500641363	71.98883707	15.45914905
9	10.57920612	0.672590051	79.71928379	6.729644596	10.57824449	0.672528914	77.71270984	6.729032882	10.58000752	0.672641002	79.7240693	6.730154385
10	10.58584221	0.703082372	84.07484243	6.733865943	10.58323766	0.702909385	78.79601871	6.732209138	10.65551077	0.707709566	86.63611128	6.77818351
11	10.53410327	0.68619968	78.48190731	6.700953773	10.53695649	0.686385541	78.49685009	6.702768761	10.53428625	0.6862116	78.47152481	6.701070168
12	12.10182522	0.775513032	76.29343621	7.698213058	12.10332533	0.775609163	76.29428904	7.699167308	12.10517207	0.775727506	76.30006422	7.700342054
13	12.08835235	0.770544521	75.73100203	7.689642694	12.09092453	0.770708479	75.78432166	7.691278911	12.0902408	0.770664896	75.77164152	7.690843977
14	12.10898215	0.753238464	77.0514128	7.702765726	12.10711408	0.753122261	77.02742043	7.701577406	12.10702548	0.753116749	76.99220966	7.701521047
15	10.57115714	0.626734126	80.92870385	6.724524482	10.56991342	0.626660389	80.90490231	6.723733327	10.56886358	0.626598147	80.90425041	6.723065498
16	10.63221944	0.694691916	61.05568081	6.763367431	10.56989434	0.690619695	80.90206548	6.723721188	10.5675233	0.690464775	80.87933724	6.72221292
17	10.41976456	0.692304081	10.74344658	6.628220635	10.41976456	0.692304081	10.74344658	6.628220635	10.57157429	0.702390536	76.65132933	6.724789835
18	10.50937676	0.687577496	75.49916002	6.685224744	10.57214071	0.691683837	77.09009774	6.725150148	10.58426998	0.692477396	77.37609424	6.732865817
19	10.57345147	0.698030198	81.1161721	6.725983949	10.57175591	0.697918262	81.09633643	6.724905369	10.57345796	0.698030627	81.11784933	6.725988081
20	10.70596512	0.698273375	79.89335536	6.81027853	10.7051972	0.69822329	79.87036388	6.809790045	10.70565851	0.698253377	79.89707791	6.810083492
21	10.69318316	0.6561805	81.02188468	6.802147674	10.69152874	0.656078978	81.02909681	6.801095264	10.69259898	0.656144652	81.04109687	6.801776065
22	10.69245057	0.654892143	80.48415762	6.801681658	10.69116886	0.654813641	80.46315489	6.800866332	10.69111598	0.654810402	80.46555837	6.800832699
23	11.31261113	0.70510722	59.05650296	7.196178195	2.954919375	0.191808147	59.11280448	1.879683313	2.954953569	0.191810367	59.09634635	1.879705064
Total	271.5026672	0.70510722	1417.356071	172.7082767	263.1350569	0.68369482	1782.273431	167.3854724	263.3773056	0.684363282	1858.492691	167.5395716

Recurrent PPO

Recurrent PPO		Time Elapsed (train)		Time Elapsed (test)		Violations						
		5989		700.8		12604						
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	10.57149274	0.612720196	75.72756847	6.724737959	10.57151937	0.612721739	73.87661354	6.7247549	10.5714274	0.612716409	73.72140582	6.724696399
1	10.57876416	0.610984233	80.89488851	6.729363455	10.578398	0.610963086	80.90691973	6.729130536	10.57845961	0.610966644	80.90874182	6.72916973

2	10.58720782	0.614305416	80.55392989	6.734734638	10.58389272	0.614113063	80.52105899	6.732625837	10.58393894	0.614115744	80.50208777	6.732655239
3	10.59258766	0.635157037	79.60667022	6.738156862	10.58947318	0.634970285	79.60320869	6.736175677	10.59038966	0.63502524	79.62872866	6.736758669
4	2.861140471	0.180000074	57.16810638	1.820028676	2.863009153	0.180117636	59.23028288	1.821217383	2.861185099	0.180002881	57.20280302	1.820057065
5	10.58898954	0.654507494	78.16550418	6.735868028	10.58877798	0.654494418	78.1761886	6.735733451	10.58899247	0.654507675	78.17715602	6.735869888
6	10.58778939	0.635000717	80.11098367	6.735104588	10.58500884	0.634833953	80.07208187	6.733335822	10.58687143	0.634945662	80.12191974	6.734520657
7	16.72212237	1.004185318	78.29526207	10.63727648	16.72212703	1.004185597	78.26851884	10.63727944	16.72198446	1.004177036	80.30099599	10.63718875
8	24.30014653	1.500511303	69.96309438	15.45780921	24.30040819	1.50052746	69.97153679	15.45797566	24.30040819	1.50052746	69.97153679	15.45797566
9	10.57897957	0.672575648	79.71133279	6.729500482	10.57827293	0.672530723	79.7036667	6.729050978	10.57827091	0.672530594	79.70190873	6.729049691
10	10.57738496	0.702520665	75.07869728	6.728486118	10.52793379	0.699236256	41.63825984	6.697029245	10.58323766	0.702909385	78.79601871	6.732209138
11	10.53411122	0.686200198	76.46217935	6.700958829	10.53418141	0.68620477	76.46522523	6.701003479	10.53418141	0.68620477	76.46522523	6.701003479
12	12.10161602	0.775499627	74.27922165	7.698079984	12.10153985	0.775494746	76.28913376	7.698031531	12.10166451	0.775502734	74.29779299	7.698110825
13	12.08861933	0.770561539	75.74134517	7.689812526	12.09014358	0.770658699	75.75522259	7.690782134	12.08861933	0.770561539	75.74134517	7.689812526
14	12.10728228	0.753132724	77.01552162	7.701684403	12.10707329	0.753119723	77.00636593	7.701551459	12.10711408	0.753122261	77.02742043	7.701577406
15	10.56888778	0.626599582	80.9219604	6.723080894	10.56869488	0.626588145	78.88246988	6.722958187	10.56874644	0.626591202	80.90484421	6.722990984
16	10.56758377	0.690468727	80.87438709	6.722251389	10.56758377	0.690468727	80.87438709	6.722251389	10.56747822	0.69046183	80.84656783	6.722184244
17	10.44768886	0.694159411	25.26720173	6.645983834	10.41976456	0.692304081	10.74344658	6.628220635	10.41941975	0.692281171	10.10393354	6.62800129
18	10.57246842	0.691705277	77.12234855	6.725358613	10.5721581	0.691684974	77.03181116	6.725161207	10.57213617	0.69168354	77.12029742	6.725147264
19	10.57181202	0.697921966	81.08345926	6.72494106	10.57189092	0.697927174	81.0568727	6.724991249	10.57169905	0.697914508	81.07224472	6.724869202
20	10.70556137	0.698247042	79.89170804	6.8100217	10.70524051	0.698226114	77.88298007	6.809817593	10.70501716	0.698211547	77.88495305	6.809675517
21	10.69154077	0.656079716	79.02092907	6.801102916	10.6914424	0.656073679	81.01707432	6.801040338	10.69152874	0.656078978	81.02909681	6.801095264
22	10.69102343	0.654804734	80.47123945	6.800773823	10.69117417	0.654813966	80.44409112	6.80086971	10.69106952	0.654807556	80.46600383	6.800803142
23	11.31281741	0.705123854	74.93163214	7.196309412	2.954953569	0.191810367	59.09634635	1.879705064	2.954886907	0.19180604	59.0790463	1.879662659
Total	271.5076179	0.705123854	1798.359171	172.7114259	263.0646622	0.683502891	1734.513763	167.3406929	263.1187271	0.683652184	1771.072075	167.3750847

A2C

A2C	Time Elapsed (train)	3726	Time Elapsed (test)	740.3	Violations	12604
------------	-----------------------------	-------------	----------------------------	--------------	-------------------	--------------

Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	10.5714853	0.612719765	73.95736956	6.724733226	10.57146782	0.612718752	73.82686431	6.724722111	10.57146782	0.612718752	73.82686431	6.724722111
1	10.57872196	0.610981796	80.90009698	6.729336613	10.57872196	0.610981796	80.90009698	6.729336613	10.57846162	0.61096676	80.90940375	6.729171005
2	10.58393894	0.614115744	80.50208777	6.732655239	10.58393894	0.614115744	80.50208777	6.732655239	10.58393894	0.614115744	80.50208777	6.732655239
3	10.58932616	0.63496147	79.60449213	6.73608216	10.58940474	0.634966182	79.59944834	6.736132141	10.58932616	0.63496147	79.60449213	6.73608216
4	2.861185099	0.180002881	57.20280302	1.820057065	2.861185099	0.180002881	57.20280302	1.820057065	2.861185099	0.180002881	57.20280302	1.820057065
5	10.58893736	0.654504269	78.18154232	6.735834836	10.58893736	0.654504269	78.18154232	6.735834836	10.58893736	0.654504269	78.18154232	6.735834836
6	10.58504333	0.634836022	80.08626119	6.73335776	10.58504333	0.634836022	80.08626119	6.73335776	10.58504333	0.634836022	80.08626119	6.73335776
7	16.72198576	1.004177114	78.29410438	10.63718958	16.72207835	1.004182674	78.30454226	10.63724848	16.72207835	1.004182674	78.30454226	10.63724848
8	24.30040819	1.50052746	69.97153679	15.45797566	24.30040819	1.50052746	69.97153679	15.45797566	24.30040819	1.50052746	69.97153679	15.45797566
9	10.57827135	0.672530622	79.70229029	6.72904997	10.57824449	0.672528914	77.71270984	6.729032882	10.57824449	0.672528914	77.71270984	6.729032882
10	10.58323766	0.702909385	78.79601871	6.732209138	10.58323766	0.702909385	78.79601871	6.732209138	10.58323766	0.702909385	78.79601871	6.732209138
11	10.53418141	0.68620477	76.46522523	6.701003479	10.53418141	0.68620477	76.46522523	6.701003479	10.53418141	0.68620477	76.46522523	6.701003479
12	12.10165943	0.775502409	74.28183146	7.698107599	12.10165943	0.775502409	74.28183146	7.698107599	12.10165943	0.775502409	74.28183146	7.698107599
13	12.08861933	0.770561539	75.74134517	7.689812526	12.08852914	0.770555791	75.75218307	7.689755157	12.08861933	0.770561539	75.74134517	7.689812526
14	12.10711408	0.753122261	77.02742043	7.701577406	12.10711408	0.753122261	77.02742043	7.701577406	12.10728251	0.753132738	77.01572439	7.701684551
15	10.56888778	0.626599582	80.9219604	6.723080894	10.56888778	0.626599582	80.9219604	6.723080894	10.56888778	0.626599582	80.9219604	6.723080894
16	10.56758377	0.690468727	80.87438709	6.722251389	10.56758377	0.690468727	80.87438709	6.722251389	10.56758377	0.690468727	80.87438709	6.722251389
17	10.41976456	0.692304081	10.74344658	6.628220635	10.41976456	0.692304081	10.74344658	6.628220635	10.41976456	0.692304081	10.74344658	6.628220635
18	10.57214071	0.691683837	77.09009774	6.725150148	10.57214071	0.691683837	77.09009774	6.725150148	10.57214071	0.691683837	77.09009774	6.725150148
19	10.57189141	0.697927207	81.09427562	6.724991561	10.57189141	0.697927207	81.09427562	6.724991561	10.57189141	0.697927207	81.09427562	6.724991561
20	10.70524051	0.698226114	77.88298007	6.809817593	10.70524051	0.698226114	77.88298007	6.809817593	10.70524051	0.698226114	77.88298007	6.809817593
21	10.69152874	0.656078978	81.02909681	6.801095264	10.69152874	0.656078978	81.02909681	6.801095264	10.69152874	0.656078978	81.02909681	6.801095264
22	10.69117414	0.654813964	78.4514334	6.800869695	10.69116975	0.654813695	80.46908396	6.800866902	10.69116975	0.654813695	80.46908396	6.800866902
23	11.31140552	0.705033043	74.29574361	7.213344988	2.954953569	0.191810367	59.09634635	7.213346048	2.954953569	0.191810367	59.09634635	7.213343726
Total	271.4737325	0.705033043	1783.097847	172.7078044	263.1173128	0.683648829	1767.812246	172.707826	263.1172325	0.683648682	1767.804063	172.7077726

DDPG

DDPG	Time Elapsed (Training) 4656				Time Elapsed (Testing) 695.7				Violations 12276			
Step	Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions
0	3.845228471	0.222868161	29.0754419	2.446026735	3.845228471	0.222868161	29.0754419	2.446026735	3.845228471	0.222868161	29.0754419	2.446026735
1	3.847576371	0.222219577	34.53617725	2.447520281	3.847576371	0.222219577	34.53617725	2.447520281	3.847576371	0.222219577	34.53617725	2.447520281
2	3.850030784	0.223391739	34.3928393	2.449081582	3.850030784	0.223391739	34.3928393	2.449081582	3.850030784	0.223391739	34.3928393	2.449081582
3	3.851856992	0.230966611	33.66808429	2.45024327	3.851856992	0.230966611	33.66808429	2.45024327	3.851856992	0.230966611	33.66808429	2.45024327
4	3.850293624	0.242229678	33.71900772	2.44924878	3.850293624	0.242229678	33.71900772	2.44924878	3.850293624	0.242229678	33.71900772	2.44924878
5	3.851462595	0.238059649	34.25524136	2.449992386	3.851462595	0.238059649	34.25524136	2.449992386	3.851462595	0.238059649	34.25524136	2.449992386
6	3.850283446	0.23092004	34.13510949	2.449242306	3.850283446	0.23092004	34.13510949	2.449242306	3.850283446	0.23092004	34.13510949	2.449242306
7	9.988584005	0.599827533	32.46944299	6.353938057	9.988584005	0.599827533	32.46944299	6.353938057	9.988584005	0.599827533	32.46944299	6.353938057
8	3.845794754	0.237474226	34.1638865	2.446386959	3.845794754	0.237474226	34.1638865	2.446386959	3.845794754	0.237474226	34.1638865	2.446386959
9	3.848255816	0.244659056	31.86221773	2.44795249	3.848255816	0.244659056	31.86221773	2.44795249	3.848255816	0.244659056	31.86221773	2.44795249
10	3.869026579	0.256970049	39.55067057	2.461165187	3.869026579	0.256970049	39.55067057	2.461165187	3.869026579	0.256970049	39.55067057	2.461165187
11	3.832655798	0.24966218	33.90994724	2.438029006	3.832655798	0.24966218	33.90994724	2.438029006	3.832655798	0.24966218	33.90994724	2.438029006
12	5.401208613	0.346121977	29.67284195	3.435816823	5.401208613	0.346121977	29.67284195	3.435816823	5.401208613	0.346121977	29.67284195	3.435816823
13	5.381077848	0.343004566	31.22899313	3.423011241	5.381077848	0.343004566	31.22899313	3.423011241	5.381077848	0.343004566	31.22899313	3.423011241
14	5.389598783	0.335259649	33.32251656	3.428431578	5.389598783	0.335259649	33.32251656	3.428431578	5.389598783	0.335259649	33.32251656	3.428431578
15	3.84444849	0.22792652	34.507839	2.445530573	3.84444849	0.22792652	34.507839	2.445530573	3.84444849	0.22792652	34.507839	2.445530573
16	3.84370692	0.251141555	34.44446066	2.445058846	3.84370692	0.251141555	34.44446066	2.445058846	3.84370692	0.251141555	34.44446066	2.445058846
17	3.768726041	0.250399556	4.057068547	2.397362009	3.768726041	0.250399556	4.057068547	2.397362009	3.768726041	0.250399556	4.057068547	2.397362009
18	3.851734899	0.252000313	32.32203922	2.450165604	3.851734899	0.252000313	32.32203922	2.450165604	3.851734899	0.252000313	32.32203922	2.450165604
19	3.848247608	0.254050728	32.91698294	2.447947269	3.848247608	0.254050728	32.91698294	2.447947269	3.848247608	0.254050728	32.91698294	2.447947269
20	3.972444368	0.259094076	34.13333118	2.526951311	3.972444368	0.259094076	34.13333118	2.526951311	3.972444368	0.259094076	34.13333118	2.526951311
21	3.854114148	0.236505305	34.57275447	2.451679092	3.854114148	0.236505305	34.57275447	2.451679092	3.854114148	0.236505305	34.57275447	2.451679092

22	3.965652097	0.242888605	32.08049964	2.522630612	3.965652097	0.242888605	32.08049964	2.522630612	3.965652097	0.242888605	32.08049964	2.522630612
23	3.930736871	0.246162662	33.59552843	2.760944608	3.930736871	0.246162662	33.59552843	2.760944608	3.930736871	0.246162662	33.59552843	2.760944608
Total	103.3827459	0.268491834	772.5929221	66.0243566	103.3827459	0.268491834	772.5929221	66.0243566	103.3827459	0.268491834	772.5929221	66.0243566

SAC

SAC		Time Elapsed (training) 6422				Time (Elapsed (testing) 704.9				Violations 12608			
Step	Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	10.57458864	0.612899634	76.27177568	6.726707328	10.57317492	0.612817695	75.88840743	6.725808032	10.5792757	0.613171294	76.90525391	6.729688855	
1	10.58061689	0.61109124	80.91899485	6.730542018	10.58490761	0.611339053	80.91588895	6.733271431	10.58213031	0.611178648	80.88758849	6.731504733	
2	10.58666824	0.614274107	80.42294162	6.734391399	10.58942605	0.614434125	80.51913208	6.736145698	10.58791294	0.614346329	80.50056266	6.735183181	
3	10.59901232	0.635542276	79.54369086	6.742243718	10.59910012	0.63554754	79.61298426	6.742299571	10.59563874	0.635339987	79.64978506	6.740097715	
4	2.866596757	0.18034334	59.22544918	1.823499529	2.834927133	0.17835094	60.74646891	1.803353848	2.866017415	0.180306892	59.21314178	1.823130998	
5	10.59087079	0.654623775	80.11359137	6.737064729	10.59138034	0.65465527	80.19701542	6.737388865	10.59397597	0.654815706	80.20004106	6.739039994	
6	10.58832577	0.635032886	79.99781452	6.73544579	10.59042785	0.635158958	80.11477931	6.736782963	10.58898914	0.635072671	80.1197081	6.735867769	
7	16.72466407	1.00433795	80.30062759	10.63889331	16.72897303	1.004596709	80.25723112	10.64163433	16.72543099	1.004384005	80.31765297	10.63938116	
8	10.57428487	0.652952193	80.16088566	6.72651409	24.30539612	1.50083546	71.97863055	15.46114858	24.30228188	1.500643159	71.93543518	15.45916755	
9	10.5811364	0.672712772	79.74067141	6.730872489	10.5837779	0.67288071	79.7280367	6.732552795	10.57959789	0.672614959	79.71961795	6.729893811	
10	10.66203313	0.708142763	85.52283411	6.782332516	10.57795108	0.702558265	75.01236199	6.728846239	10.11255888	0.671648203	77.32423537	6.432800956	
11	10.53927873	0.686536814	78.50194833	6.704245984	10.53568959	0.686303014	78.48497445	6.701962863	10.54135025	0.686671754	78.44331024	6.70556372	
12	12.10526337	0.775733357	76.28737868	7.700400136	12.10357553	0.775625196	76.27402418	7.699326464	12.1058853	0.775773212	76.28446907	7.700795756	
13	12.0926839	0.770820626	75.77791045	7.69239808	12.09225711	0.770793421	75.78400491	7.692126593	12.09271407	0.770822549	75.80283859	7.692417276	
14	12.11668201	0.753717433	76.72684761	7.707663762	12.10944538	0.753267279	77.03120176	7.703060396	12.11094834	0.753360771	77.04072186	7.704016458	
15	10.57494772	0.626958858	80.90233511	6.726935741	10.57672187	0.627064043	80.84749141	6.728064313	10.57611289	0.627027938	80.85525991	6.727676931	
16	10.57650022	0.691051313	80.48818897	6.727923318	10.56911132	0.690568534	80.87462227	6.723223096	10.65809786	0.69638277	77.89960679	6.779829212	
17	10.56872546	0.702201256	74.42445968	6.722977637	10.50718853	0.698112654	53.72882047	6.683832767	10.57042415	0.70231412	74.42557375	6.724058213	

18	10.61892762	0.694744877	78.24439838	6.75491224	10.51102465	0.68768531	75.31026235	6.686272999	10.57473694	0.691853695	77.06813293	6.72680166
19	10.57409126	0.698072435	81.13851543	6.726390935	10.57543904	0.698161412	81.14954537	6.727248283	10.57434627	0.69808927	81.14207921	6.726553152
20	10.71033402	0.698558327	79.80786611	6.813057674	10.70864093	0.698447899	79.87545981	6.811980669	10.70709801	0.698347266	79.88886414	6.810999186
21	10.6934557	0.656197224	81.03204264	6.802321042	10.69606922	0.656357601	80.97474606	6.803983553	10.6922168	0.6561212	81.03059121	6.801532948
22	10.69868339	0.655273892	80.47647264	6.805646481	10.69460193	0.65502391	80.51664604	6.80305018	10.69405766	0.654990575	80.50703535	6.802703961
23	10.73036397	0.669209537	78.52294091	6.825799128	2.957615474	0.191983155	59.11207539	1.881398355	2.957811124	0.191995855	59.11018012	1.881522812
Total	257.5287353	0.669209537	1884.550582	163.8191791	263.1968227	0.683857006	1824.934811	167.4247629	262.9696095	0.683219701	1846.271686	167.280228

TQC

TQC	Time Elapsed (train) 6760				Time Elapsed (Test) 699.5				Violations 12584			
Step	Episode 0 Water Saved	Episode 0 Water Saved %	Episode 0 Rewards	Episode 0 Carbon Emissions	Episode 1 Water Saved	Episode 1 Water Saved %	Episode 1 Rewards	Episode 1 Carbon Emissions	Episode 2 Water Saved	Episode 2 Water Saved %	Episode 2 Rewards	Episode 2 Carbon Emissions
0	10.58142172	0.613295677	77.48546615	6.731053985	10.58161552	0.613306909	77.57560299	6.731177263	10.57589309	0.612975239	79.82109227	6.727537115
1	10.58761627	0.611495493	82.25676146	6.734994459	10.5850734	0.611348629	81.89609602	6.733376892	10.58665851	0.611440177	82.12519781	6.734385209
2	10.58274739	0.614046607	80.7053703	6.731897272	10.58605583	0.614238573	81.01491263	6.734001834	10.59455522	0.614731737	82.07736832	6.739408467
3	10.59432549	0.635261242	79.63751861	6.739262328	10.58918841	0.63495321	80.29880917	6.735994531	10.59017672	0.635012472	80.35423275	6.736623218
4	2.868864468	0.180486006	59.26908357	1.824942065	2.871657453	0.180661718	60.88592096	1.826718739	2.86993753	0.180553514	60.63836011	1.825624662
5	10.57978072	0.653938295	79.74895992	6.730010109	10.59753436	0.655035651	80.13438461	6.741303555	10.57841461	0.653853856	79.67032826	6.729141104
6	10.59518882	0.635444496	79.9334325	6.739811509	10.57874633	0.634458361	79.98194236	6.729352115	10.59049431	0.635162944	80.10441179	6.736825242
7	16.73051608	1.004689371	80.23726662	10.64261589	16.7322013	1.004790571	79.95464027	10.64368789	16.72947442	1.004626818	81.44542992	10.64195327
8	10.58037671	0.653328359	81.59342807	6.730389235	10.56671828	0.652484963	80.19399716	6.721700833	24.29423899	1.500146517	72.01935434	15.4540513
9	10.5722429	0.672147353	79.95258114	6.725215151	10.58566801	0.673000876	81.26191101	6.733755132	10.57588687	0.672379025	80.29309723	6.727533158
10	10.65958383	0.707980087	84.7262579	6.780774468	10.55460264	0.701007527	66.07151151	6.713993831	10.66431528	0.708294337	87.81657635	6.783784238
11	10.52947796	0.685898384	78.64751561	6.698011523	10.52812666	0.685810359	78.51169144	6.69715193	10.54324918	0.686795452	78.26317815	6.706771669
12	12.1104791	0.776067593	76.15308298	7.703717965	12.1094169	0.775999525	76.22903918	7.703042281	12.11103025	0.776102913	76.08228307	7.704068566
13	12.09778793	0.771145971	75.61376028	7.695644856	12.09693122	0.771091362	75.78096333	7.695099886	12.09648394	0.771062851	77.32902044	7.694815364

14	12.10448668	0.752958823	77.25788156	7.699906065	12.10612203	0.75306055	77.4014147	7.700946344	12.10028418	0.752697408	76.92287966	7.697232775
15	10.57777771	0.627126641	82.42311627	6.728735956	10.56894518	0.626602985	81.38494702	6.723117405	10.57817737	0.627150336	80.68096067	6.728990191
16	10.52505821	0.687690176	-287.9438089	6.695200031	10.61713245	0.693706156	-288.1165191	6.753770292	10.55794259	0.689838787	80.59496518	6.716118438
17	10.55614262	0.701365235	66.37173456	6.714973445	10.51929872	0.698917273	60.29960753	6.691536303	10.49074462	0.697020097	36.33300411	6.673372465
18	10.57554231	0.691906387	76.79396556	6.727313976	10.56454578	0.691186937	77.44317826	6.72031886	10.51213403	0.687757891	76.8852724	6.6869787
19	10.57513711	0.698141479	82.46694831	6.727056219	10.57487912	0.698124447	82.44108438	6.726892105	10.54516157	0.696162577	79.13943843	6.707988181
20	10.70973781	0.698519441	79.86176575	6.812678415	10.71010589	0.698543448	79.80779268	6.812912558	10.70871276	0.698452584	79.88815423	6.812026358
21	10.69595804	0.656350779	80.97675458	6.803912826	10.6958612	0.656344836	80.73371136	6.803851228	10.68272349	0.655538649	81.3424602	6.795494064
22	10.69314651	0.654934768	80.51572875	6.802124356	10.68790175	0.654613537	81.18783055	6.798788061	10.69502255	0.655049672	81.80614685	6.803317747
23	10.69961412	0.669934996	80.45141636	6.806238532	2.964240457	0.192413192	58.80412332	1.88561264	2.961152922	0.192212776	59.09505076	1.883648597
Total	257.3830105	0.668923069	1515.135988	163.7264806	249.5725689	0.6488209	1471.178593	158.7581025	263.232865	0.68395911	1830.728263	167.4476901

ARS

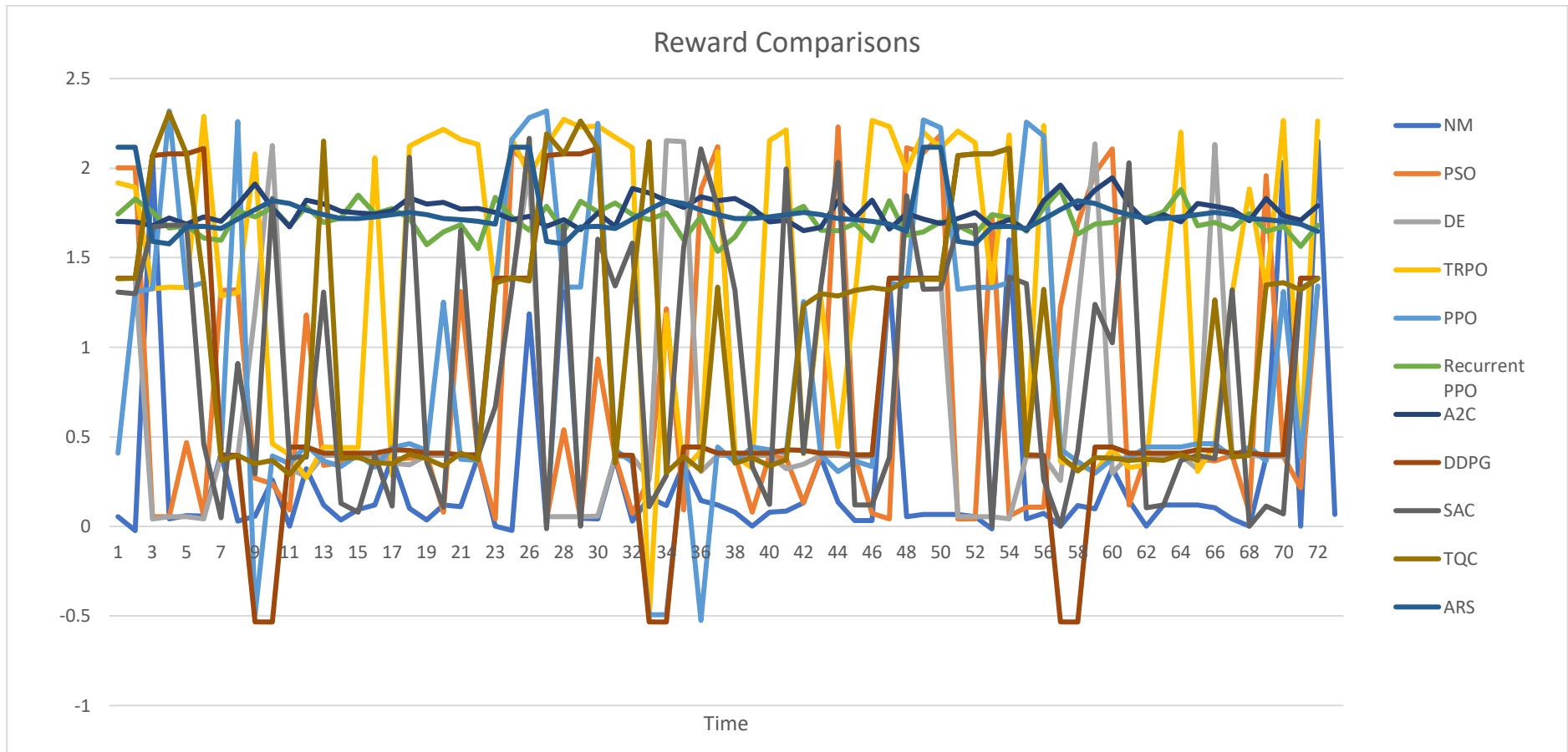
ARS		Time Elapsed (train) 3535				Time Elapsed (test) 691.6				Violations 12604			
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	
	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	Water Saved	Water Saved %	Rewards	Carbon Emissions	
0	10.57146782	0.612718752	73.82686431	6.724722111	10.57146782	0.612718752	73.82686431	6.724722111	10.57146782	0.612718752	73.82686431	6.724722111	
1	10.57872196	0.610981796	80.90009698	6.729336613	10.57872196	0.610981796	80.90009698	6.729336613	10.57872196	0.610981796	80.90009698	6.729336613	
2	10.58393894	0.614115744	80.50208777	6.732655239	10.58393894	0.614115744	80.50208777	6.732655239	10.58393894	0.614115744	80.50208777	6.732655239	
3	10.58932616	0.63496147	79.60449213	6.73608216	10.58932616	0.63496147	79.60449213	6.73608216	10.58932616	0.63496147	79.60449213	6.73608216	
4	2.861185099	0.180002881	57.20280302	1.820057065	2.861185099	0.180002881	57.20280302	1.820057065	2.861185099	0.180002881	57.20280302	1.820057065	
5	10.58893736	0.654504269	78.18154232	6.735834836	10.58893736	0.654504269	78.18154232	6.735834836	10.58893736	0.654504269	78.18154232	6.735834836	
6	10.58504333	0.634836022	80.08626119	6.73335776	10.58504333	0.634836022	80.08626119	6.73335776	10.58504333	0.634836022	80.08626119	6.73335776	
7	16.72207835	1.004182674	78.30454226	10.63724848	16.72207835	1.004182674	78.30454226	10.63724848	16.72207835	1.004182674	78.30454226	10.63724848	
8	24.30040819	1.50052746	69.97153679	15.45797566	24.30040819	1.50052746	69.97153679	15.45797566	24.30040819	1.50052746	69.97153679	15.45797566	
9	10.57824449	0.672528914	77.71270984	6.729032882	10.57824449	0.672528914	77.71270984	6.729032882	10.57824449	0.672528914	77.71270984	6.729032882	

10	10.58323766	0.702909385	78.79601871	6.732209138	10.58323766	0.702909385	78.79601871	6.732209138	10.58323766	0.702909385	78.79601871	6.732209138
11	10.53418141	0.68620477	76.46522523	6.701003479	10.53418141	0.68620477	76.46522523	6.701003479	10.53418141	0.68620477	76.46522523	6.701003479
12	12.10165943	0.775502409	74.28183146	7.698107599	12.10165943	0.775502409	74.28183146	7.698107599	12.10165943	0.775502409	74.28183146	7.698107599
13	12.08861933	0.770561539	75.74134517	7.689812526	12.08861933	0.770561539	75.74134517	7.689812526	12.08861933	0.770561539	75.74134517	7.689812526
14	12.10711408	0.753122261	77.02742043	7.701577406	12.10711408	0.753122261	77.02742043	7.701577406	12.10711408	0.753122261	77.02742043	7.701577406
15	10.56888778	0.626599582	80.9219604	6.723080894	10.56888778	0.626599582	80.9219604	6.723080894	10.56888778	0.626599582	80.9219604	6.723080894
16	10.56758377	0.690468727	80.87438709	6.722251389	10.56758377	0.690468727	80.87438709	6.722251389	10.56758377	0.690468727	80.87438709	6.722251389
17	10.41976456	0.692304081	10.74344658	6.628220635	10.41976456	0.692304081	10.74344658	6.628220635	10.41976456	0.692304081	10.74344658	6.628220635
18	10.57214071	0.691683837	77.09009774	6.725150148	10.57214071	0.691683837	77.09009774	6.725150148	10.57214071	0.691683837	77.09009774	6.725150148
19	10.57189141	0.697927207	81.09427562	6.724991561	10.57189141	0.697927207	81.09427562	6.724991561	10.57189141	0.697927207	81.09427562	6.724991561
20	10.70524051	0.698226114	77.88298007	6.809817593	10.70524051	0.698226114	77.88298007	6.809817593	10.70524051	0.698226114	77.88298007	6.809817593
21	10.69152874	0.656078978	81.02909681	6.801095264	10.69152874	0.656078978	81.02909681	6.801095264	10.69152874	0.656078978	81.02909681	6.801095264
22	10.69116975	0.654813695	80.46908396	6.800866902	10.69116975	0.654813695	80.46908396	6.800866902	10.69116975	0.654813695	80.46908396	6.800866902
23	10.69597967	0.669706263	79.79372028	6.803926586	2.954953569	0.191810367	59.09634635	1.879705064	2.954953569	0.191810367	59.09634635	1.879705064
				0								
Total	270.8583505	0.703561201	1788.503826	172.2984139	263.1173244	0.683648872	1767.806452	167.3741924	263.1173244	0.683648872	1767.806452	167.3741924

Appendix H: Burst Leakage – Jowitt & Xu Results

This section shows the episodic performance of each of the optimisation algorithms as they tackle the background leakage case study on the Jowitt & Xu network. The results and discussions associated with these results were covered in section 4.2 of the thesis.

Each algorithm’s step rewards, water saved% and carbon emissions are listed along with the algorithm’s processing speeds. A line plot of the rewards of the algorithms across the three test episodes is shown below.



NM

Step	Episode 0		Episode 0	Episode 0	Episode 0	Episode 1		Episode 1	Episode 1	Episode 1	Episode 2		Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0	11.61422965	0.017568206	0.053231412	2	7.388043762	4.968009116	0.007514834	0.022647722	0	3.160249959	14.50430899	0.021939869	0.066356362	1	9.226481035
1	4.872685871	0.007353453	0.022160501	0	3.099612936	262.1677518	0.395641812	1.188005705	1	166.7701502	14.36609036	0.021680111	0.06555616	1	9.138557399
2	474.6435087	0.68942709	2.069414242	1	301.9302288	-5.98798E-06	-8.69764E-09	-2.62881E-08	0	-3.80908E-06	12.71753296	0.018472415	0.055665024	0	8.089877065
3	9.41631329	0.013684174	0.041260776	0	5.98990521	341.9219591	0.496895072	1.491516177	0	217.5033966	2.979524515	0.004329968	-0.01302992	0	1.895335135
4	13.942366	0.020261597	0.061059257	2	8.869017859	10.54341115	0.015322102	0.046196931	2	6.706874704	366.7548206	0.532982592	1.600013828	2	233.3000765
5	12.81246589	0.018646132	0.056198524	0	8.150265802	9.415751968	0.013702855	0.041324502	0	5.989548142	9.415718279	0.013702806	0.041324354	0	5.989526712
6	511.395869	0.8013255	0.406121401	13	325.3091402	506.1406452	0.793090892	0.380995068	11	321.9661872	15.34678405	0.024047456	0.072959859	1	9.762396269
7	6.44634742	0.010073787	0.030358829	0	4.100650521	6.4463214	0.010073746	0.030358706	0	4.100633969	-3.21586E-06	-5.02546E-09	-1.53754E-08	0	-2.04567E-06
8	10.9384082	0.019156981	0.057843685	0	6.958140225	29.88941394	0.052346823	0.159375985	1	19.013254	21.74700933	0.038086623	0.116596173	2	13.83370757
9	429.2962178	0.752241009	0.260287876	11	273.0839101	21.75866519	0.038126961	0.116724617	2	13.8411221	18.29977709	0.03206607	0.098148035	1	11.6408542
10	2.47165E-05	4.19929E-08	1.27934E-07	0	1.57226E-05	460.0941791	0.781692168	0.348961498	13	292.6751092	455.5715853	0.774008358	0.325005175	11	289.7981968
11	454.7467684	0.773347975	0.323041191	11	289.2735143	28.23198556	0.048011664	0.145967891	1	17.95893066	28.23198872	0.048011669	0.145967907	1	17.95893267
12	24.74647402	0.039819116	0.12073531	1	15.74172706	24.74082734	0.03981003	0.120707675	0	15.73813509	-8.77913E-05	-1.41263E-07	-4.28144E-07	0	-5.58458E-05
13	7.622812725	0.012265076	0.036977348	0	4.84902363	16.34860479	0.02630484	0.079978899	1	10.39967448	24.7428569	0.039811159	0.120710868	1	15.73942613
14	19.33674976	0.031126278	0.094650551	2	12.30049326	0.000130554	-2.10153E-07	-6.36805E-07	0	-8.30482E-05	24.7721263	0.039875579	0.120908753	1	15.75804498
15	24.7449143	0.039815834	0.12072524	1	15.74073488	16.35004595	0.026308061	0.079988918	1	10.40059123	24.745166	0.039816239	0.120726471	1	15.740895
16	479.0155389	0.793557927	0.383496956	12	304.7113646	17.14487414	0.028402942	0.086553225	1	10.90619734	20.30967223	0.033645884	0.102544887	2	12.9193887
17	20.21981896	0.033398315	0.101766646	2	12.86223124	26.45770672	0.043701817	0.132676866	1	16.8302764	8.670563514	0.014321702	0.043195463	0	5.515518863
18	7.55018603	0.012126212	0.036557772	0	4.802824337	494.7153782	0.794553084	0.385714392	11	314.6983464	3.84015E-08	6.16759E-11	7.3147E-10	0	2.4428E-08
19	24.77680357	0.039885882	0.120940403	1	15.76102029	27.93429618	0.044968837	0.13640274	2	17.76956449	493.6979982	0.794758697	0.386365392	11	314.0511706
20	23.03482033	0.036168163	0.109532525	1	14.65290991	6.638125344	0.010422864	0.03141283	0	4.222644294	431.1265872	0.676934158	2.032790275	2	274.2482447

21	508.0533983	0.796207549	0.390509111	12	323.1829278	6.561308265	0.010282705	0.030989602	0	4.173779413	-1.53095E-05	-2.39927E-08	-7.34156E-08	-9.73869E-06
22	0.000201074	-3.03438E-07	-9.22891E-07	0	0.000127907	526.7734798	0.794945199	1.386284409	11	335.091146	474.6807165	0.716332866	2.15056324	301.9538974
23	519.0486171	0.783166925	1.350933027	10	88.75265802	11.61376236	0.017523435	0.053078811	2	66.75018542	14.35606466	0.02166116	0.06549777	57.30981329
Total	3588.529766	23.85798339	6.203480787	82	2041.311007	2846.920348	18.64422023	6.450567061	61	1870.345411	2471.077736	16.24093864	7.717865561	42 1620.079603

PSO

PSO																	
Time Elapsed 3438																	
Step	Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2		
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions		
0	440.776132	2	0.666737782	2.00159306	8	1	280.3865132	1	0.699537509	9	1	294.1799433	9	0.694979748	2.08633182	1	292.2632456
1	441.696264	0.666571342	2.00107479	1	280.9718275	2	0.673922655	6	1	284.0705386	8	0.728338657	7	2	307.0078633		
2	12.7157496	0.018469824	8	0	8.088742636	9	0.013671088	1	0	5.987166388	6	0.013664629	7	0	5.984337612		
3	12.7415775	0.018516585	8	0	8.105172328	4	0.17932436	8	0	78.49475599	1	0.013683834	9	0	5.989756079		
4	107.214085	0.155807744	5	0	68.20102399	6	0.013674688	1	0	5.985759671	1	0.577754091	1	1	252.8977037		
5	12.8036895	0.01863336	3	0	8.144683013	1	0.31152552	5	0	136.1684955	9	0.018643861	9	0	8.149273074		
6	492.741931	0.772095941	7	10	313.4429974	4	0.796197697	0.39047736	12	323.2274377	8	0.035807145	3	0	14.5364045		
7	494.612316	0.772936789	9	10	314.6327866	3	0.023795195	0.07217843	0	9.686107088	22.6789211	0.035440631	5	0	14.42651529		

	431.029613		0.26865953			430.094543		0.26373536			422.560595		1.22414036		
8	5	0.754883686	2	11	274.1865577	7	0.753246052	9	11	273.5917411	6	0.740051472	6	7	268.7992461
	424.789643		0.23698870			420.780330		1.21594073			321.075811		1.69137631		
9	8	0.744344294	5	8	270.2171882	9	0.737318913	4	7	267.6667841	5	0.56261011	1	0	204.2427452
	17.6972362		0.09180543			17.6977237		0.09180796			386.593473		1.97361515		
10	4	0.030067303	8	0	11.25756591	9	0.030068131	9	0	11.25787606	2	0.656815721	3	1	245.9198402
	230.852402		1.18063208			368.197717					412.594700		2.10814965		
11	7	0.392590449	2	0	146.8498304	8	0.62616159	1.8816029	1	234.2179323	1	0.701663647	6	3	262.4597406
			0.33978062			438.837772		2.12063720			24.7243058		0.12062681		
12	484.228794	0.779164037	2	11	308.0276204	7	0.706126143	1	3	279.153484	7	0.039783446	6	0	15.72762545
	487.693477		0.35637630			496.258783		0.39772856					0.39774252		
13	1	0.784696876	5	11	310.2315746	4	0.79847842	8	11	315.6801373	496.261621	0.798482985	4	11	315.6819424
	496.074280		0.39788613			16.3577661		0.08006118			496.087913		0.39795322		
14	1	0.798528507	3	11	315.562771	9	0.026331022	4	0	10.40550223	9	0.798550453	6	12	315.5714438
	496.227497					496.245070					496.247824		0.39775865		
15	1	0.798455441	0.39765867	11	315.6602355	1	0.798483717	0.39774511	11	315.671414	2	0.798488148	8	11	315.6731659
	476.948878		0.37321770			477.717906		0.37704059			478.056329		0.37864585		
16	5	0.790134208	5	11	303.3967206	9	0.791408213	5	11	303.8859149	4	0.791968859	6	11	304.1011923
	479.646754		0.37954491			26.4556768		0.13266664			477.045706		0.36665657		
17	9	0.792261966	6	11	305.1128937	3	0.043698464	3	1	16.82898514	5	0.787965655	1	11	303.4583148
	497.039590		0.39712161			496.025261		0.39214232			497.051188		0.39717855		
18	5	0.798285957	3	11	316.1768243	7	0.796656863	7	11	315.5315895	7	0.798304585	2	12	316.1842021
	16.3528313		0.08004301			461.258070		2.22989281			16.3616997		0.08008644		
19	4	0.026324909	1	0	10.40236307	5	0.742536661	1	6	293.4154838	2	0.026339185	4	1	10.40800443
	278.017478		1.31127358			507.062746		0.39039658			415.519119				
20	8	0.436529626	8	0	176.8524786	8	0.79616545	1	11	322.5527545	1	0.652428064	1.95913868	1	264.3200221
	508.001229		0.39025963			15.3493022		0.07298367			507.788287		0.38924128		
21	8	0.796125792	3	11	323.1497423	4	0.024055011	8	0	9.763998138	9	0.795792075	7	11	323.0142857
	9.07122449							0.04148569			47.9149583		0.21754299		
22	3	0.013689236	0.04148081	0	5.770387325	9.07229362	0.013690849	1	0	5.771067417	8	0.072307676	4	0	30.47966332
	9.06063420		0.04142520			466.938062					515.151135		1.33328629		
23	2	0.013671145	2	0	212.2299142	8	0.704539873	2.11500088	2	186.7010819	9	0.777286208	1	10	198.4916759
Tota	7358.03331		13.5611671			6933.00557		18.3427432			7427.83511		19.7946606		
I	3	49.33134499	7	129	4887.058415	9	46.25255868	8	100	4299.89595	6	49.65479536	8	106	4595.78821

DE

DE		Time Elapsed																						
		2967																						
Step	Episode 0		Episode 0		Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2	
	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations
0	525.388604	0.794726409	1.385640173	11	334.2101988	522.8609257	0.790902929	1.374165491	11	332.6022921	525.3587944	0.794681318	1.385503025	10	334.1912363									
1	526.7541365	0.794933623	1.386249164	10	335.0788413	439.850863	0.663786416	1.992719805	1	279.797931	526.6553939	0.794784609	1.385795875	10	335.0160292									
2	9.388227865	0.013636547	0.041114654	0	5.97203951	12.66818952	0.018400743	0.055449055	0	8.058488716	12.69099319	0.018433865	0.055548864	0	8.072994586									
3	12.72200881	0.018488147	0.055715022	0	8.092724245	12.7343784	0.018506123	0.05576919	0	8.100592789	12.66969813	0.018412127	0.055485938	0	8.059448374									
4	12.67511261	0.018419974	0.05550958	0	8.062892634	12.7254699	0.018493156	0.055730109	0	8.094925912	12.73386495	0.018505356	0.055766872	0	8.100266172									
5	9.371352808	0.01363824	0.041129748	0	5.961304948	12.74294222	0.018544954	0.055893584	0	8.106040407	9.410274062	0.013694883	0.041300474	0	5.986063536									
6	508.1051369	0.796169128	0.390390183	11	323.2158397	508.1074402	0.796172737	0.390401196	11	323.2173049	508.0995986	0.79616045	0.390363702	11	323.2123167									
7	509.3290635	0.795934832	0.389644778	11	323.9944039	509.375784	0.796007843	0.389867525	11	324.0241237	509.3966224	0.796040407	0.389966873	11	324.0373794									
8	416.9410277	0.730209642	1.194585104	7	265.2245265	430.6565888	0.754230389	0.266648097	11	273.9492692	428.940364	0.751224679	0.257674825	10	272.8575444									
9	403.5445187	0.707117191	2.125222445	6	256.7027393	408.9217161	0.71653947	2.153496117	6	260.123282	422.7768847	0.740817405	1.226462468	8	268.9368319									
10	454.5759004	0.772316706	0.320276911	11	289.1648218	420.9027729	0.715106637	2.148584533	6	267.7446719	418.6000409	0.711194334	2.136847855	6	266.279858									
11	445.7461006	0.758041328	0.277449635	11	283.5480095	451.2155112	0.767342676	0.305358406	11	287.027211	449.7778491	0.76489777	0.298037337	11	286.1126854									
12	496.2533102	0.798512474	0.397833224	12	315.6766557	496.2389252	0.798489328	0.397762462	11	315.6675051	496.2033256	0.798432045	0.397587346	11	315.6448595									
13	496.2658615	0.798489808	0.397763383	11	315.6846398	496.2761692	0.798506393	0.397814085	12	315.6911967	496.2628213	0.798484917	0.397748429	11	315.6827059									
14	496.0841395	0.798544377	0.397934652	11	315.5690428	496.0854823	0.798546539	0.39794126	12	315.569897	496.0706581	0.798522676	0.397868309	11	315.560467									
15	496.1981713	0.798408254	0.397514418	11	315.6415807	496.2386514	0.798473389	0.397713538	11	315.6673309	496.2256364	0.798452447	0.397649517	11	315.6590518									
16	472.6411443	0.782997828	0.351800523	11	300.6564847	466.8703596	0.773437696	0.32311993	11	296.9855731	465.9490526	0.771911419	0.318544821	11	296.3995113									
17	472.4752983	0.780416431	0.343995939	11	300.5509867	473.1043225	0.781455429	0.347125003	11	300.9511216	430.0037208	0.710263521	2.133419764	2	273.5339669									
18	497.0472372	0.798298239	0.397159155	12	316.1816885	497.0466986	0.798297374	0.397156511	12	316.1813459	497.0217831	0.798257357	0.397034194	11	316.1654967									
19	495.9542684	0.798390858	0.397466032	11	315.4864292	496.0555023	0.798553825	0.397964246	11	315.5508261	495.9982591	0.798461674	0.397682527	11	315.5144126									
20	507.0394464	0.796128865	0.390290232	11	322.5379327	507.1406216	0.796287725	0.390775057	11	322.6022922	507.1335473	0.796276618	0.390741161	11	322.5977921									
21	508.0287224	0.796168878	0.390391107	11	323.1672309	508.0438736	0.796192622	0.390463562	11	323.1768689	508.0201146	0.796155388	0.390349942	11	323.1617553									

22	526.7668733	0.794935229	1.386253844	10	335.0869435	526.7733551	0.794945011	1.386283591	10	335.0910667	526.7362382	0.794888998	1.386113224	10	335.0674558
23	526.8387575	0.794921085	1.386209404	10	253.6536825	526.863486	0.794958397	1.386322892	10	250.8586405	526.86646	0.794962884	1.38633654	11	252.3083033
Total	9826.134421	65.6243504	14.29753931	210	6169.12164	9729.500029	65.00907415	15.85452524	201	6104.839798	9779.601995	65.30798811	16.06982988	199	6138.158432

TRPO

Step	Time Elapsed (training)					Time Elapsed (testing)					Episode				
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0	422.025351	0.63837450	1.91684638		268.458766	476.237772	0.72037865	2.16288261		302.944371	484.332522	0.73262313	2.19960224		308.093604
	3	9	2	3	5	2	2	2	4	7	6	6	2	4	3
1	417.369917	0.62986003	1.89126449		265.497352	432.205870	0.65224922	1.95844809		274.934798	466.124841	0.70343692	2.11203109		296.511334
	7	9	8	3	265.497352	7	8	3	3	5	5	4	6	3	2
2	533.605387	0.77507013	1.32657890		339.437058	488.596688	0.70969430	2.13037953		310.806125	506.278445	0.73537732	2.20745793		322.053844
	2	7	3	10	9	2	1	9	3	3	6	9	9	3	8
3	535.838907	0.77870316	1.33748585		340.857845		0.75696927	2.27226134			491.148600	0.71375737			312.429447
	2	6	4	10	7	520.883445	3	3	7	331.344377	5	2	2.14257656	3	8
4	534.421253	0.77664207			339.956047	510.946418	0.74252751	2.22891985		325.023235	535.834559	0.77869594	1.33746418		340.855080
	3	5	1.33129643	10	6	3	8	3	4	6	9	8	5	10	2
5	524.022090	0.76261551			333.340932	511.574518	0.74450041	2.23484943		325.422782	500.239714	0.72800474	2.18534289		318.212487
	9	5	2.2892151	8	5	9	6	8	4	9	3	2	1	3	1
6	485.964808	0.76147661	1.28682236		309.131933	461.536425	0.72319885	2.17202314		293.592550	515.972677	0.80849707	0.42786290		328.220539
	4	1	5	10	9	4	5	4	5	9	7	5	4	16	7
7	490.926328		1.30389373		312.288055	450.323429	0.70372599	2.11355086				0.74477314	2.23668865		303.168451
	2	0.76717665	5	10	9	6	8	7	4	286.45974	476.590033	4	3	8	8

	395.029460		2.08062716		251.286140	471.586111	0.82591230	0.51715866		299.985357	448.741368	0.78590316	0.36270359		285.453359
8	3	0.69183482	7	9	3	3	6	6	21	1	5	7	3	20	3
	467.127695	0.81853180	0.46073574		297.149269	414.330624	0.72601731	1.18325288		263.563996	444.413556	0.77873060	0.34139235		282.700351
9	6	7	3	20	7	3	4	6	12	7	5	2	9	20	5
	469.729135	0.79806179			298.804097	452.930859	0.76952181	0.31286413		288.118378	451.670634	0.76738070	0.30644180		287.316724
10	1	4	0.39845907	20	4	9	1	1	17	6	7	9	7	17	2
	444.453426		0.27185854		282.725713	473.642556	0.80548238	0.42073387		301.293502			0.43043111		302.516622
11	1	0.75584299	4	13	4	1	5	8	20	8	475.565337	0.80875229	3	20	2
	505.467563	0.81333896	0.44292711		321.538026	432.991397	0.69671884	2.09307316		275.434487	481.752520	0.77517950	0.32849199		306.452413
12	7	7	1	17	6	3	3	6	4	7	6	9	6	16	4
	504.852686	0.81230597	0.43977061		321.146891	495.463260	0.79719842	0.39452707		315.174089	484.978457	0.78032841	0.34393373		308.504496
13	9	6	8	17	2	3	6	1	16	1	2	9	3	16	2
	504.967583	0.81284401	0.44141881		321.219979	481.136632	0.77448344	0.32641260		306.060634	472.019250	0.75980723	1.28238734		300.260885
14	6	7	7	17	3	8	4	2	16	8	7	5	8	10	7
	425.871022	0.68524827	2.05867936			445.678943	0.71712023	2.15432404		283.505289	455.029240	0.73216533	2.19946178		289.453200
15	8	3	8	4	270.905075	8	1	5	5	7	7	8	6	8	6
	459.115493	0.76059064	0.28541235		292.052547	444.818043	0.73690487	2.21435742		282.957653	463.217061	0.76738548	0.30579414		294.661637
16	6	8	4	14	8	4	2	1	9	7	6	4	9	15	2
	427.599538	0.70629238	2.12240741		272.004618	484.546223	0.80035471			308.229543	495.606378	0.81862345	0.45934744		315.265129
17	8	6	1	5	6	2	8	0.40456635	18	5	3	5	2	20	4
	450.159303	0.72299240	2.17188542			473.394661	0.76031028	1.28385185		301.135812	476.486627	0.76527622	1.29874744		303.102673
18	3	8	4	6	286.355336	8	1	4	10	3	5	9	7	11	5
	458.373431	0.73789294				505.296893	0.81343068	0.44321197		321.429460	389.349942	0.62677842	1.88323949		247.673285
19	1	7	2.2166549	8	291.580507	8	5	3	17	1	1	3	6	3	2
	458.213040	0.71946399	2.16085722		291.478479	481.625925	0.75622578	1.27113725		306.371883	493.518039	0.77489820	1.32712592		313.936695
20	3	9	4	4	2	3	1	4	10	6	8	5	9	11	5
	453.317309	0.71042663	2.13367698		288.364206		0.75413172	2.26481227		306.104228	481.528167	0.75463792	2.26633795		306.309698
21	5	1	2	4	9	481.205164	2	5	8	9	8	5	9	9	1
		0.76800061	1.30569791		323.733265	492.739478	0.74358504	2.23246137		313.441437	531.158924	0.80156320			337.880814
22	508.918546	2	6	10	5	5	7	2	6	1	2	1	0.40636555	13	9
	465.574925	0.70686598	1.78915365	5.66666666	299.799040	438.183034	0.66115282				499.570548	0.75377742	2.26301547		303.325089
23	3	8	1	7	1	1	5	1.98515481	3	300.44965	9	6	4	8	2
	11342.9442	74.6268873	33.4636255	237.6666666	7219.11118	11321.8743	74.5491455	35.7408973		7223.78338	11521.1274	75.8181386	30.6542436		7314.35786
Total	1	5	7	7	7	8	5	1	226	8	5	9	5	267	6

PPO

PPO		Time Elapsed (train)	292	Time Elapsed (Test)	26.6										
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	s	Carbon Emissions	Water Saved	Water Saved %	Rewards	s	Carbon Emissions
0	530.183831	0.80197988	0.40765276		337.260538	475.882884	0.71984183	2.16127465		302.718620	499.770341	0.75597507	2.26965476		317.913909
	6	6	2	13	9	7	4	7	4	6	8	5	4	9	9
1	509.319502	0.76862272	1.30756336		323.988322	503.863049	0.76038829	2.28286917		320.517363	490.998543		2.22463720		312.333993
	8	1	4	10	1	5	5	5	9	1	6	0.74097425	3	6	6
2	533.951355	0.77557266	1.32808797		339.657136	531.907164	0.77260344	2.31917147		338.356785	532.886332	0.77402569			338.979653
	7	2	3	10	4	4	2	6	9	4	7	9	1.32344248	10	9
3	531.506207	0.77240670	2.31858630		338.101728	535.838908	0.77870316	1.33748585		340.857846	535.838907	0.77870316	1.33748585		340.857845
	1	8	1	9	5	1	7	8	10	2	2	6	4	10	7
4	534.840029	0.77725065	1.33312399		340.222439	535.834560	0.77869594	1.33746418		340.855080	534.525239	0.77679319	1.33175023		340.022195
	8	8	6	10	7	6	9	8	10	7	9	3	4	10	6
5	540.359939	0.78639217	1.36056734		343.733764	515.416519	0.75009172	2.25162980		327.866756	540.521567	0.78662739	1.36127371		343.836579
	2	7	9	12	5	6	5	7	5	5	5	7	8	12	5
6	515.972676	0.80849707	0.42786289		328.220538	515.693518		0.42652935			479.690035	0.75164443	2.25735575		305.140425
	2	3	7	16	8	6	0.80805965	7	16	328.042961	8	3	4	8	6
7	481.604283	0.75260897	2.26019802		306.358116		0.78532665	0.35832918		319.676222	464.448698		2.17977449		295.445106
	7	5	1	8	9	502.540751	2	3	13	5	6	0.72579973	2	5	1
			-												
8	476.327221	0.83421564	0.49227559		303.001272	476.327221	0.83421564			303.001272	461.424920	0.80811650	0.42949619		293.521620
	6	9	1	21	2	6	9	-0.49227559	21	2	5	5	2	20	4
								-							
9	454.149962	0.79579137	0.39255695				0.83435958	0.49182830		302.895184	448.592631	0.78605345	0.36335458		285.358744
	2	1	3	20	288.893874	476.160448	9	6	21	2	2	1	2	20	6
10	460.402681	0.78221630	0.35093938		292.871353	473.596290	0.80463202	0.41816040		301.264072	449.895968	0.76436558			286.187823
	2	8	3	18	6	6	5	7	20	4	9	2	0.2973419	17	7

	479.781066	0.81592161	0.45203570			484.232354	0.82349152	0.52527469		308.029885		0.79160701	0.37912901		296.103370
11	4	2	2	20	305.198332	9	8	7	21	6	465.48351	1	5	20	4
	489.648504	0.78788479	0.36659738		311.475206	505.467563	0.81333896	0.44292711		321.538026	492.716041	0.79282071	0.38140009		313.426528
12	6	8	9	16	7	7	7	1	17	6	2	7	4	16	1
	482.873143	0.77694097	0.33372059		307.165264	487.810920	0.78488584	0.35759511		310.306282	505.487915	0.81332805	0.44289322		321.550973
13	8	8	4	16	2	6	2	6	16	8	9	7	5	17	1
		0.79853789	0.39850808		315.566481	505.090379	0.81304168	0.44202273		321.298092	505.172104	0.81317323	0.44242466		321.350079
14	496.080113	6	9	16	5	6	1	1	17	3	9	4	2	17	4
	482.599127	0.77652669	0.33250475		306.990956	502.460679		0.42837225		319.625287		0.81333502	0.44291485		
15	1	6	6	16	7	5	0.80848495	3	16	4	505.474923	2	7	17	321.542708
	489.952555	0.81167666	0.43862229		311.668619	483.721433	0.80135391	0.40766955		307.704878	494.853141	0.81979518	0.46296311		314.785980
16	2	3	6	19	4	5	6	3	18	3	9	7	5	20	6
	495.893422	0.81909758	0.46079898		315.447724	454.235030	0.75028786	1.25440622		288.947987	495.893422	0.81909758	0.46079898		
17	7	4	9	20	1	5	2	6	12	6	7	3	9	20	315.447724
	502.560828	0.80715351	0.42433630					0.39956481		316.416669	494.963037	0.79495084	0.38770187		314.855887
18	2	4	9	16	319.688994	497.416634	0.79889152	3	16	2	5	5	6	16	4
	465.618430	0.74955600	1.25164266		296.189196	476.781564	0.76752649	0.30553202			505.296893	0.81343068	0.44321197		321.429460
19	9	2	4	9	3	8	7	8	14	303.290289	8	5	3	17	1
	503.707184	0.79089670	0.37512445			501.956636	0.78814808	0.36688149		319.304655	501.574777	0.78754850	0.36508337		319.061747
20	1	8	5	15	320.418214	3	2	3	15	5	3	5	7	15	3
	503.485017	0.78904810	0.36954400		320.276889	495.928047	0.77720502	0.33400500		315.469749	491.326322	0.76999332	1.31236088		312.542500
21	9	7	8	15	6	2	7	8	13	4	6	8	4	11	3
	517.276374	0.78061327	1.34353752		329.049847	521.333561	0.78673590	1.36190418			526.816644	0.79501033	0.38672616		335.118604
22	6	4	2	11	4	3	7	3	12	331.630705	7	8	8	13	1
	506.595605	0.76437718	2.29483197		316.927043		0.77889442	1.33837947		316.271998		0.78024981	1.34244534		
23	9	9	6	9	8	516.216993	7	8	11	6	517.115286	7	5	11	315.077039
Total	11984.6890	78.8491050	19.8366681		7618.37185	11975.7131	78.8300174	18.8227955		7605.88667	11940.7672	78.5559117	22.6256207		7581.8905
	7	2	6	345	5	2	3	1	336	2	1	2	5	337	

Recurrent PPO

	Time Elapsed (train)	Time Elapsed (test)
Recurrent PPO	529	19.9

Step	Episode 0		Episode 0		Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 1		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2		Episode 2		Episode 2			
	Water	Water Saved	Rewards	Violations	Carbon	Water Saved	Water	Rewards	Violations	Carbon	Water Saved	Water	Rewards	Violations	Carbon	Water Saved	Water	Rewards	Violations	Carbon	Water Saved	Water	Rewards	Violations	Carbon	Water Saved	Water	Rewards	Violations	Carbon		
	Saved	%		s	Emissions		Saved %		s	Emissions		Saved %		s	Emissions		Saved %		s	Emissions		Saved %		s	Emissions		Saved %		s	Emissions		
	383.906697		1.74381231		244.210728		0.57420790			241.474470		0.54749922			361.948277		0.54749922			1.64412673											230.242538	
0	2	0.58071452	8	3	2	379.6052167	7	1.72428514	3	4	361.948277	6	8	9	2	3														3		
	403.125967		1.82675838		256.436490		0.55016840		1.65210647						375.215299		0.56624378			1.70034964											238.681956	
1	5	0.608364252	6	3	4	364.5631186	1	2	2	231.905891	3	8	1	3	2																2	
	402.631503		1.75568044		256.121951		0.59547289								384.691178		0.55876992			1.67747614											244.709752	
2	3	0.58482853	4	3	9	409.9597282	1	1.78762411	3	3	384.691178	4	9	9	2																4	
	381.826658		1.66582537		242.887574		0.54857238		1.64687634						373.870127		0.54332346			1.63112320											237.826265	
3	8	0.554886224	2	2	2	377.4819992	6	7	2	3	373.870127	8	6	9	2																7	
	384.791622		1.67874917		244.773646		0.60466739		1.81522105						398.893370		0.57968760															253.744050
4	1	0.559194385	3	2	6	416.082413	8	6	3	5	398.893370	2	2	1.74025716	3																6	
	368.165106		1.60854256		234.197187		0.58447960		1.75465185						394.887840		0.57468492			1.72525780												251.196053
5	3	0.535795011	9	2	4	401.6181419	7	5	3	4	394.887840	8	1	5	3																3	
	339.411464		1.59776281		215.906420		0.60093631		1.80515469						349.855631		0.54820200			1.64688317												222.550164
6	3	0.531836642	9	2	6	383.5100097	8	6	3	4	349.855631	5	2	7	2																3	
	375.664970		1.76344971		238.968001		0.57977802		1.74160592						378.510849																	240.778321
7	9	0.5870563	7	3	3	371.0075069	3	3	3	3	378.510849	5	0.59150359	1.77679691	3																6	
	327.726418		1.72680113		208.473329		0.56939531		1.71308364						357.193418		0.62557067			1.88174896												227.217877
8	1	0.573963641	3	2	1	325.1179571	4	3	2	9	357.193418	4	1	6	4																3	
	338.077369		1.78217806		215.057776		0.58184047		1.75046798						309.254950		0.54189682			1.63052383												196.723259
9	9	0.592401356	8	2	5	332.0503829	4	5	2	6	309.254950	4	1	1	2																1	
	328.441739		1.67809213		208.928359		0.53187755		1.59960980						330.128997		0.56088353			1.68669873												210.001657
10	5	0.558016917	4	2	3	313.0564377	1	6	2	1	330.128997	4	9	9	2																8	
			1.78383189		221.899410		0.57515943		1.72958386						331.671962		0.56404543			1.69618637												210.983168
11	348.832626	0.593229075	3	3	1	338.2072555	9	2	2	4	331.671962	1	8	5	2																6	
	350.656749		1.69551328		223.059771		0.51078403		1.53506405						358.766882		0.57728548			1.73468145												228.218789
12	5	0.564235609	2	2	5	317.4380806	3	2	2	8	358.766882	6	1	4	3																3	
	356.072600				226.504902		0.53810172		1.61706629						356.269095		0.57323556			1.72252637												226.629897
13	7	0.572919406	1.7215774	2	7	334.4332208	9	6	2	4	356.269095	8	6	4	2																2	
	382.273757		1.84892162		243.171982		0.58661939		1.76270714						363.241459		0.58470812			1.75697048												231.065157
14	6	0.615344325	8	3	7	364.4288092	2	3	3	1	363.241459	8	2	8	3																4	
	361.303963		1.74690256		229.832677		0.57205852		1.71899410						389.071977		0.62603672			1.88100366												247.496466
15	7	0.58135657	9	3	4	355.5253745	4	4	2	2	389.071977	9	6	6	3																6	
	356.110784		1.77333360		226.529192		0.57878968		1.73983663						337.092106		0.55844141			1.67875250												214.431030
16	8	0.589948578	3	3	5	349.3749402	6	7	2	222.244387	337.092106	6	1	4	2																8	

	346.156654		1.71869068		220.197171		0.59426635	1.78622552		228.861583	341.828349	0.56461885	1.69722913		217.443849
17	8	0.571768179	1	2	2	359.7773746	8	9	3	6	8	3	3	2	9
	326.020919		1.57354731		207.388427		0.54984550	1.65228156		217.777106	343.572687	0.55180564	1.65816527		218.553458
18	5	0.52361608	5	2	3	342.3522395	6	9	2	6	9	5	1	2	2
	340.018584		1.64488071		216.292621		0.54917333	1.65031050		217.007411	360.969338	0.58109111	1.74611545		229.619815
19	4	0.547364438	8	2	9	341.1422554	2	4	2	5	3	7	7	3	5
	356.957695		1.68376449		227.067929		0.56238093	1.68947672		227.838975	349.197370	0.54829285	1.64719241		222.131431
20	8	0.560477744	4	2	5	358.1698034	7	3	2	3	2	6	2	2	1
	329.407611		1.55094731		209.542769		0.53019473	1.59283723		215.207563	355.931162	0.55780569	1.67571055		226.414931
21	2	0.516238702	1.83736648	2	7	338.3128394	5	2	2	4	6	4	8	2	1
	405.477404		1.83736648		257.932286		0.60579086	1.81903547		255.357419	344.838289	0.52038979	1.56273107		219.358532
22	7	0.611899286	1	3	7	401.429635	3	7	3	4	6	4	5	2	8
	380.251146		1.73305490		228.065832		0.54140504	1.62580290		371.377180	0.56035275	1.68266880		228.484540	
23	6	0.577087172	6	3	8	358.8194678	4	7	2	227.650436	7	9	6	2	6
	8673.31001		41.1399844		5503.44644		56.7331910	40.9099090		5491.31744	8618.27780	56.6932292		5474.50296	
Total	7	57.05226226	1	58	1	8633.464208	4	7	57	8	5	4	40.8811759	58	6

A2C

A2C		Time Elapsed (train)					Time Elapsed (test)									
		362					17.4									
Step	Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 1		Episode 2		Episode 2	
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	
0	375.1403511	0.567454151	1.70401602	3	238.634280	377.113571	0.57043893	1.71297386	3	239.889485	377.525323	0.57106176	1.71484309	3	240.151408	
1	374.7858033	0.565595628	1.698406937	3	238.408745	381.707455	0.57604121	1.72975573	3	242.811746	372.836059	0.56265323	1.68957626	2	237.168474	
2	384.5633341	0.558584233	1.676918846	2	244.628428	383.905985	0.55762942	1.67405331	2	244.210275	393.975499	0.57225554	1.71794818	3	250.615694	
3	394.2980358	0.573010143	1.720217648	3	250.820866	392.348675	0.57017725	1.71171587	3	249.580839	401.848008	0.58398207	1.75314505	3	255.623555	

					245.995585	379.456691	0.55144145	1.65548696		241.379990	383.679752	0.55757857	1.67390555		244.066364
4	386.7125478	0.56198595	1.687132763	2	9	2	2	9	2	4	7	2	1	2	3
					251.653870	400.716213		1.75071305		254.903597	391.593361	0.56989042	1.71086909		249.100369
5	395.607543	0.575732312	1.728401105	3	2	2	0.58316702	8	3	5	4	6	4	3	1
					230.026166	356.015044	0.55785341	1.67585112		226.468290	351.346131	0.55053752	1.65389309		223.498300
6	361.6081351	0.566617443	1.702155207	2	9	8	9	2	2	3	1	2	5	2	9
					243.601169	401.750435	0.62782037	1.88578606		255.561486	387.412726	0.60541466	1.81854600		246.440983
7	382.9484525	0.598438287	1.797609064	3	6	3	8	3	3	9	1	3	1	3	3
					230.702151	353.621416	0.61931484	1.86296765		224.945655	361.465985		1.90421319		229.935742
8	362.6708034	0.635163488	1.910547742	4	4	8	6	3	4	6	1	0.63305343	1	4	4
					214.393807	345.028564	0.60458169			219.479570	336.727313	0.59003569	1.77507504		214.198978
9	337.0335905	0.590572378	1.77668646	2	6	2	5	1.81874932	3	3	7	9	7	2	8
					208.298493	348.417570	0.59195551	1.77998348		221.635384	367.488161	0.62435612	1.87724809		233.766569
10	327.4515711	0.556334638	1.673041313	2	4	1	3	5	3	7	3	2	1	3	2
					226.624259	359.727572	0.61175715	1.83945326		228.829903	380.620481	0.64728789	1.94610726		242.120300
11	356.2602335	0.605860557	1.82175199	3	8	3	6	1	3	3	8	6	5	4	9
					236.591661	375.822742	0.60472976	1.81705605			371.577831	0.59789935			236.368090
12	371.929292	0.598464883	1.798252796	3	2	5	6	6	3	239.068363	8	4	1.79655542	3	3
					231.407546	378.683239	0.60929983	1.83077125		240.887982	350.669888	0.56422646			223.068129
13	363.7797061	0.585320108	1.758798419	3	6	7	5	4	3	5	5	4	1.69548474	2	5
					229.902594	367.685535	0.59186172	1.77844182		233.892122	359.895175	0.57932162	1.74080285		228.936519
14	361.4138752	0.581766267	1.748140501	3	3	1	9	5	3	6	7	3	3	3	1
					229.460907	351.255438	0.56518797	1.69837153		223.440609		0.56545369	1.69916910		223.545656
15	360.7195299	0.580416186	1.744079994	3	4	7	9	6	2	7	351.420575	2	2	2	2
					225.103906	343.329842	0.56877511	1.70977393		218.398979	362.169232	0.59998526	1.80346110		230.383092
16	353.8701918	0.586236715	1.762191367	3	4	1	4	9	2	2	8	6	1	3	4
					234.923250	332.513597	0.54923310	1.65104152		211.518549	359.845142	0.59437829	1.78656152		228.904691
17	369.3064996	0.610006198	1.833470959	3	5	2	4	3	2	4	3	4	8	3	9
					237.291202	345.487613	0.55488117	1.66739695		219.771580	366.182431	0.58811873	1.76716125		232.935968
18	373.0289922	0.599114863	1.800165172	3	5	1	1	8	2	4	6	1	5	3	4
					238.016828	376.004412	0.60529469	1.81876095		239.183926	352.540894	0.56752294	1.70538976		224.258313
19	374.1696978	0.602341153	1.809896315	3	1	6	1	4	3	9	2	5	3	2	6
					238.952208	365.147565	0.57333708	1.72236002		232.277669	387.800096	0.60890499	1.82910719		246.687397
20	375.6401438	0.589812022	1.771805927	3	3	1	2	7	3	1	1	2	2	3	1
					239.966308	386.709051		1.82047414		245.993361	368.279566		1.73379280		234.269998
21	377.2343409	0.591191459	1.775911376	3	9	4	0.60603997	8	3	8	9	0.57715778	9	3	1
					246.170739	366.103837	0.55248128	1.65904771		232.885973	377.285726	0.56935568	1.70969122		239.998996
22	386.9878946	0.58399707	1.753631815	3	5	9	3	4	2	3	7	8	4	3	5

23	375.2586427	0.566209306	1.700246347	3	8	385.674161	0.58192477	1.74741059	3	8	245.335047	394.835467	0.59574781	1.78889433	3	3	251.162737
						5650.28450	8854.22623	58.2717699	42.0183962		5632.35039	8909.02083	58.6507483	42.2914412			5667.20633
Total	8882.419208	58.45927266	42.15347608	68	6	3	7	5	65	1	2	3	6	67	2		

DDPG

DDPG		Time Elapsed (Training)					Time Elapsed (Testing)								
		631					17.8								
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0		0.79511586			334.373978		0.79511586	1.38679676		334.373978		0.79511586	1.38679676		334.373978
1	525.646071	5	1.386796765	10	7	525.646071	5	5	10	7	525.646071	5	5	10	7
2	526.9041899	2	1.386914336	10	3	526.904189	2	6	10	3	526.904189	2	6	10	3
3	474.6435079	9	2.069414238	1	2	474.643507	9	8	1	2	474.643507	9	8	1	2
4	476.846214	5	2.080056581	1	7	476.846214	5	1	1	7	476.846214	5	1	1	7
5	476.8411321	9	2.080032008	1	9	476.841132	9	8	1	9	476.841132	9	8	1	9
6	483.0421613	1	2.110086742	1	6	483.042161	1	2	1	6	483.042161	1	2	1	6
7	509.8924412	3	0.398817097	14	7	509.892441	3	7	14	7	509.892441	3	7	14	7
8	511.0909331	8	0.397933221	14	4	511.090933	8	1	14	4	511.090933	8	1	14	4
		0.82094122	-		298.179776		0.82094122	0.53303897		298.179776		0.82094122	0.53303897		298.179776
	468.747684	7	0.533038977	19	7	468.747684	7	7	19	7	468.747684	7	7	19	7

SAC

SAC		Time Elapsed (training)				Time (Elapsed (testing)												
		860				18												
Step	Episode																	
	Episode 0			Episode 0			Episode 1			Episode 1			Episode 2			Episode 2		
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions			
0	508.54197		1.3092338			513.39534		1.3311869			511.59801	0.77386614	1.3231883					
	53	0.769243442	52	10	323.4937213	04	0.77658486	88	10	326.5810439	44	4	74	10	325.4377289			
1	507.15085		1.2975640			478.20989	0.72167468	2.1665495			513.90294	0.77553967	1.3281332					
	06	0.765349971	42	10	322.6087991	68	6	59	3	304.1988795	5	2	14	11	326.9039413			
2	383.13684		1.6704730			2.9548008	0.00429189	0.0129151			383.13684		1.6704730					
	54	0.556512236	16	1	243.7210101	35	4	42	0	-1.879607907	77	0.55651224	26	1	243.7210116			
3	385.15803		1.6801244			385.15802	0.55972750	1.6801244			385.15802	0.55972750	1.6801244					
	39	0.559727516	76	1	245.0067285	76	7	48	1	245.0067245	88	8	54	1	245.0067253			
4	385.15336		1.6801021			8.72993E-		3.83608E-			-5.73049E-	-8.32778E-	-2.51024E-					
	08	0.559720078	48	1	245.0037559	06	1.26867E-08	08	0	5.55328E-06	07	10	09	0	-3.64528E-07			
5	105.28355		0.4600422			367.02943	0.53414224	1.6033273			318.89301	0.46408875	1.3930475					
	22	0.153220392	54	0	66.97297325	01	8	86	0	233.4747611	28	4	41	0	202.8542233			
6	10.308772		0.0490040			497.57738	0.77967280	1.3407646			500.65001		1.3551821					
	12	0.016153204	27	0	6.557616118	91	2	67	10	316.5189288	11	0.78448741	74	10	318.4734851			
7	193.82456		0.9101962			337.21153	0.52696464	1.5822510			55.124678	0.08614401	0.2593966					
	51	0.302892047	7	2	123.2956824	74	4	97	0	214.5070032	1	7	26	2	35.06591023			
8	435.58042		0.2931770			20.664417	0.03619062	0.1107843			-2.71262E-	-4.75075E-	-1.45974E-					
	49	0.762853749	65	13	277.0814199	95	7	9	2	13.14504955	06	09	08	0	-1.72555E-06			
9	347.59790		1.8315551			433.79813		0.2846563			460.10717	0.80622998	0.4227609					
	76	0.609083868	49	3	221.113981	87	0.76012957	02	18	275.947672	15	4	98	18	292.683374			
10	467.41357		0.3853979			305.28141	0.51866791	1.5581889			438.79877	0.74551163	1.2394544					
	91	0.794127704	78	11	297.331126	27	6	83	0	194.1956122	34	6	76	8	279.1286757			
11	467.45670		0.3882920			412.29518	0.70115428	2.1072545			200.45928	0.34090354	1.0241791					
	91	0.794962657	26	18	297.3585618	28	4	04	5	262.2692117	88	4	2	0	127.5161628			
12	477.79776		1.3088141			368.18188	0.59243500	1.7792757			419.70811	0.67534494	2.0281953					
	96	0.768815988	31	13	303.9367172	81	1	96	1	234.2078627	38	5	67	2	266.9847253			
13	26.719199		0.1303834			479.40834	0.77136612	1.3161615			21.576404		0.1052163					
	45	0.042991086	75	2	16.99661715	37	9	53	10	304.9612356	62	0.03471635	03	0	13.72518251			

14	16.277242		0.0796668			68.950721	0.11098966	0.3344386			24.585971	0.03957592	0.1199973		
	93	0.026201404	6	0	10.35427977	86	3	3	0	43.86093319	28	6	14	0	15.63962805
	496.06548		0.3972238			24.837310	0.03996450	0.1211774			489.83948	0.78817680	0.3671747		
15	69	0.798194758	53	13	315.5571776	44	4	32	2	15.79950992	55	3	26	13	311.5966935
	22.832828		0.1147793			400.62599	0.66369440	1.9942975			481.76634	0.79811503	0.3976028		
16	78	0.037825855	51	0	14.52441904	63	9	54	3	254.8462088	66	4	66	13	306.4612084
	415.43212		2.0608837			485.29200	0.80158657	0.4079508			479.56647	0.79212936	0.3788842		
17	74	0.686194727	71	2	264.2646849	49	2	97	13	308.7039502	4	1	55	11	305.0618254
	489.32469		0.3599718			484.29114		1.3354758			481.16927	0.77279694	1.3204294		
18	09	0.785895202	01	13	311.2692224	58	0.77781092	94	10	308.0672837	88	8	37	10	306.0814017
	21.975746		0.1072262			420.36858	0.67671246	2.0320544			0.0001353	-2.17895E-	-6.60441E-		
19	55	0.03537672	22	0	13.9792119	58	6	71	1	267.4048648	55	07	07	0	-8.61018E-05
	350.45207		1.6521516			24.902908	0.03910134	0.1184368			23.998832	0.03768180	0.1141274		
20	91	0.550262939	81	1	222.9295766	06	4	53	2	15.84123787	57	9	33	2	15.26613738
	508.35114		0.3919798			25.275315		0.1199729			14.511095	0.02274139	0.0687797		
21	24	0.796674166	21	13	323.3723287	02	0.03961079	39	2	16.07813339	35	6	3	0	9.230797973
	147.01533		0.6660300			526.06342	0.79387367	0.3833046			513.84953	0.77544189	1.3278981		
22	36	0.221858423	93	0	93.51939402	88	3	22	13	334.6394683	82	1	73	11	326.8699682
	526.99987		1.3869256			407.46355	0.61480171	1.8455313			526.36822	0.79421112	1.3841984		
23	79	0.795164192	03	10	203.0331641	55	5	38	1	188.8062047	56	2	72	12	179.4017614
Tot	7695.8501		20.611198			7463.3271	49.3023518	25.540251			7244.7683	47.5997594	19.308443		
al	02	50.78875968	97	137	4763.282169	86	6	2	107	4677.182177	99	6	4	135	4453.11048

TQC

TQC	Time Elapsed (train)					Time Elapsed (Test)									
	Episode 0 Water Saved	Episode 0 Water Saved %	Episode 0 Rewards	Episode 0 Violations	Episode 0 Carbon Emissions	Episode 1 Water Saved	Episode 1 Water Saved %	Episode 1 Rewards	Episode 1 Violations	Episode 1 Carbon Emissions	Episode 2 Water Saved	Episode 2 Water Saved %	Episode 2 Rewards	Episode 2 Violations	Episode 2 Carbon Emissions
0	524.698220	0.79368210			333.771031	525.975224	0.79561375	1.38834983		334.583359	524.204485	0.79293525	1.38025276		
	4	4	1.38254067	10	9	3	7	9	10	7	1	8	6	10	333.456957

1	526.495311	0.79454302	1.38518595		334.914197	522.863102	0.78906159	1.36861243		525.062572	0.79238085	1.37857371		334.002803	
	6	6	6	12	6	9	8	6	10	332.603677	7	6	2	10	8
2	474.643507	0.68942708	2.06941423		301.930228	502.338009	0.72965378	2.19019809		319.547254	474.643507	0.68942708	2.06941423		301.930227
	8	9	8	1	2	9	4	2	1	8	3	8	6	1	9
3	529.800776	0.76992830	2.31113506		337.016869		0.69297255	2.08005657		476.846214	0.69297255	2.08005658			
	4	6	8	9	9	476.846213	4	6	1	303.331413	6	6	3	1	303.331414
4	476.841132		2.08003201		303.328181	518.537348	0.75355895	2.26197837		329.851978	476.841132	0.69296436	2.08003200		303.328181
	7	0.69296437	1	1	3	8	8	3	5	3	4	9	9	1	1
5	540.552310	0.78667213	1.36140807		343.856135	483.042160	0.70297693	2.11008673		307.272778	483.042161	0.70297694	2.11008674		307.272779
	3	7	4	12	6	1	9	6	1	9	4	1	2	1	7
6	503.282491	0.78861234	0.36774443		320.148058	499.840870	0.78321953			317.958774	509.218340	0.79791344	0.39564832		323.923970
	2	2	9	14	3	7	7	0.35203057	13	7	3	8	6	14	6
7	510.673435	0.79803569	0.39610808			496.416492	0.77575619	1.32895260		315.780459	495.313511	0.77403255	1.32378081		315.078831
	9	9	6	14	324.849586	3	7	8	10	1	8	6	1	10	1
8	446.469316		0.34939983		284.008061	407.596600		2.14668224		259.280349	454.912036	0.79671016	0.39378947		289.378644
	9	0.78192401	9	13	8	5	0.71384428	6	11	5	3	6	6	15	5
9	449.618056	0.78785026	0.36720339		286.011038	436.744726	0.76529277	0.29947781		277.822055	438.693088	0.76870682	0.30993491		279.061447
	9	9	6	15	3	5	5	8	11	4	9	3	7	11	7
10	448.710584	0.76235163	0.29007399		285.433777	468.330670	0.79568582	0.39007466		297.914506		0.79350136	0.38351801		297.096617
	8	5	7	11	2	7	7	5	11	3	467.044925	7	9	11	7
11	470.223517	0.79966792	0.40278949		299.118583	452.779637	0.77000264	0.31321746		288.022182	466.150098	0.79274061	0.38125342		296.527400
	4	6	3	13	9	3	8	4	11	9	1	9	6	11	4
12	445.255957	0.71645353	2.15134853		283.236219	483.359830	0.77776580	1.33540900		307.474855	490.355131	0.78902181	0.36935893		311.924706
	9	1	1	2	9	1	2	6	10	1	8	9	3	11	4
13	492.939670	0.79313798	0.38173918		313.568783	486.966480	0.78352714	0.35265550		309.769117		0.79052433	0.37394399		312.535471
	9	1	2	14	5	8	1	2	11	8	491.315273	1	1	13	5
14	493.141914	0.79380829	0.38406191		313.697434		0.79412321	0.38470275		313.821885	489.710749		0.36718531		311.514801
	8	2	3	16	8	493.337555	3	5	14	5	1	0.78828516	5	14	7
15	487.372394	0.78420712	0.35469649		310.027327		0.77851759	0.33787122		307.778035	496.502538	0.79889799	0.39912871		315.835194
	5	7	5	11	6	483.836439	4	2	14	6	4	6	8	14	7
16	472.387666	0.78257790	0.35054197		300.495242	476.927040	0.79009802	0.37283861		303.382828	476.171242	0.78884594	0.36908091		302.902050
	7	6	1	11	6	2	9	8	11	8	6	2	7	11	9
17		0.79939247	0.40134141		307.858967	449.923258	0.74316584	1.23222183		286.205183	456.679549	0.75432562			290.502995
	483.963666	3	1	13	2	4	3	6	7	2	9	5	1.26542342	8	3
18	493.107298	0.79197037	0.37796475		313.675414	476.600028	0.76545836	1.29899432		303.174810	495.200108				315.006692
	7	7	7	11	8	8	1	4	10	3	3	0.7953316	0.38824583	11	9
19	483.298093	0.77801685	0.33637423		307.435583	473.130532	0.76164903	1.28725530		300.967794	496.038928	0.79852714	0.39791776		315.540283
	7	4	2	14	3	6	9	3	11	4	6	4	2	14	3

20	510.924193	0.80222850	0.40889796		325.009097	491.677054	0.77200757	1.31795410		312.765607	498.009936	0.78195116	1.34778893		316.794080
	4	7	7	16	9	3	8	2	10	8	7	5	2	12	9
21	504.103033	0.79001664	0.37177345		320.670021	495.636052	0.77674742	1.33196056		315.284005	501.902049	0.78656732	1.36142446		319.269931
	1	4	9	11	4	7	1	6	10	9	6	2	5	10	8
22	520.655889	0.78571324	1.35883273		331.199624		0.77329866	1.32123564		325.966538	512.429320	0.77329866			325.966539
	2	3	8	12	2	512.429319	2	4	9	4	3	4	1.32123565	9	2
23			1.32951313		311.366356	524.038579	0.79069603			306.572405	525.898269	0.79350202	1.38193731		308.918885
	514.312784	0.77602126	3	11	5	8	5	1.37351634	10	2	1	8	9	10	7
Total	11803.4712	77.6633462	20.9701210		7492.62582	11639.1732	76.5612232	28.1763326		7377.13185	11722.1851	77.1930868	23.6290122		7431.10091
	2	8	6	267	4	3	2	4	222	7	7	3	7	233	7431.10091

ARS

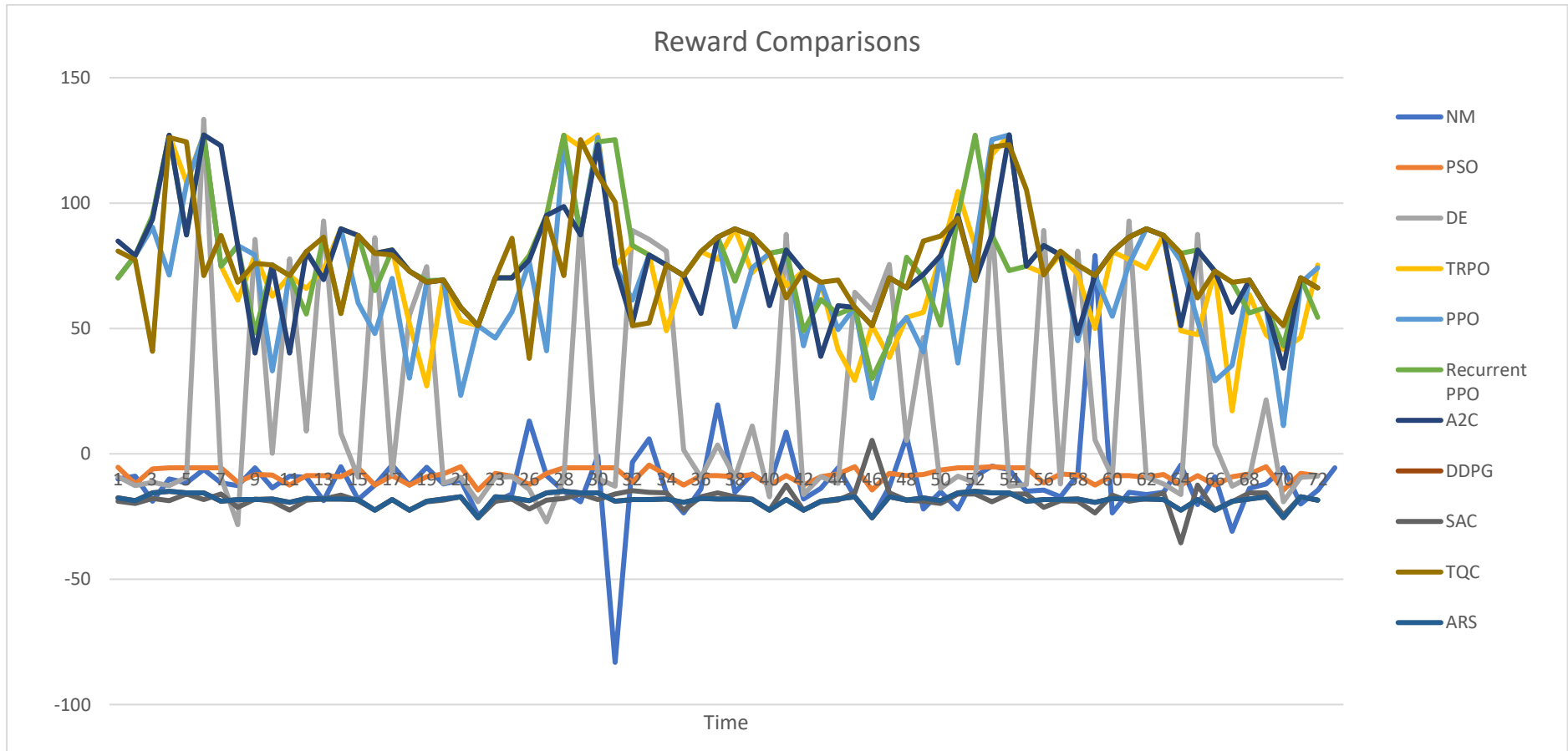
ARS	Time Elapsed (train)					Time Elapsed (test)									
	254					24.6									
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0	465.9786565	0.704860253	2.116169244	3	296.418343	465.9786565	0.704860253	2.116169244	3	296.418343	465.9786565	0.704860253	2.116169244	3	296.418343
1	467.1543951	0.70499064	2.116534444	3	297.1662538	467.1543951	0.70499064	2.116534444	3	297.1662538	467.1543951	0.70499064	2.116534444	3	297.1662538
2	364.8123153	0.529895571	1.590745163	2	232.06441	364.8123153	0.529895571	1.590745163	2	232.06441	364.8123153	0.529895571	1.590745163	2	232.06441
3	361.5571041	0.525429673	1.577344607	2	229.9937051	361.5571041	0.525429673	1.577344607	2	229.9937051	361.5571041	0.525429673	1.577344607	2	229.9937051
4	382.9380865	0.556500754	1.670599875	2	243.5945756	382.9380865	0.556500754	1.670599875	2	243.5945756	382.9380865	0.556500754	1.670599875	2	243.5945756
5	383.5751729	0.558221462	1.675773814	2	243.999839	383.5751729	0.558221462	1.675773814	2	243.999839	383.5751729	0.558221462	1.675773814	2	243.999839
6	353.1437919	0.553354344	1.662081238	2	224.6418289	353.1437919	0.553354344	1.662081238	2	224.6418289	353.1437919	0.553354344	1.662081238	2	224.6418289
7	365.8011994	0.571642062	1.716925566	3	232.693459	365.8011994	0.571642062	1.716925566	3	232.693459	365.8011994	0.571642062	1.716925566	3	232.693459
8	335.8332005	0.588161454	1.768960714	3	213.6302155	335.8332005	0.588161454	1.768960714	3	213.6302155	335.8332005	0.588161454	1.768960714	3	213.6302155
9	345.2177986	0.604913284	1.819270129	3	219.5999461	345.2177986	0.604913284	1.819270129	3	219.5999461	345.2177986	0.604913284	1.819270129	3	219.5999461
10	352.8711247	0.599522026	1.802293655	3	224.4683798	352.8711247	0.599522026	1.802293655	3	224.4683798	352.8711247	0.599522026	1.802293655	3	224.4683798

11	345.4021548	0.587395174	1.765912104	3	219.7172187	345.4021548	0.587395174	1.765912104	3	219.7172187	345.4021548	0.587395174	1.765912104	3	219.7172187
12	360.1747302	0.579550824	1.741184804	3	229.1143494	360.1747302	0.579550824	1.741184804	3	229.1143494	360.1747302	0.579550824	1.741184804	3	229.1143494
13	355.4596591	0.571933186	1.718319362	3	226.1149983	355.4596591	0.571933186	1.718319362	3	226.1149983	355.4596591	0.571933186	1.718319362	3	226.1149983
14	355.2970863	0.5719201	1.718288257	3	226.0115825	355.2970863	0.5719201	1.718288257	3	226.0115825	355.2970863	0.5719201	1.718288257	3	226.0115825
15	357.3384464	0.57497585	1.727452637	3	227.3101325	357.3384464	0.57497585	1.727452637	3	227.3101325	357.3384464	0.57497585	1.727452637	3	227.3101325
16	349.3686105	0.5787792	1.739453964	3	222.2403605	349.3686105	0.5787792	1.739453964	3	222.2403605	349.3686105	0.5787792	1.739453964	3	222.2403605
17	353.2342829	0.583458732	1.753437945	3	224.6993921	353.2342829	0.583458732	1.753437945	3	224.6993921	353.2342829	0.583458732	1.753437945	3	224.6993921
18	360.6428628	0.579221733	1.740161978	3	229.4121379	360.6428628	0.579221733	1.740161978	3	229.4121379	360.6428628	0.579221733	1.740161978	3	229.4121379
19	355.5280911	0.572331756	1.719525169	3	226.1585293	355.5280911	0.572331756	1.719525169	3	226.1585293	355.5280911	0.572331756	1.719525169	3	226.1585293
20	363.1420107	0.570188057	1.712639898	3	231.0018958	363.1420107	0.570188057	1.712639898	3	231.0018958	363.1420107	0.570188057	1.712639898	3	231.0018958
21	361.9916874	0.567303585	1.703950799	3	230.2701522	361.9916874	0.567303585	1.703950799	3	230.2701522	361.9916874	0.567303585	1.703950799	3	230.2701522
22	372.4993925	0.56213271	1.687837112	3	236.9543136	372.4993925	0.56213271	1.687837112	3	236.9543136	372.4993925	0.56213271	1.687837112	3	236.9543136
23	515.8227981	0.778299646	1.336314444	10	234.105532	363.5502359	0.548543067	1.647048898	2	234.105532	363.5502359	0.548543067	1.647048898	2	234.105532
					5621.38155			41.8919113		5621.38155	8832.51209	58.1051062	41.8919113		5621.38155
Total	8984.784658	59.06242531	41.58117692	74	1	8832.512096	58.10510624	7	66	1	6	4	7	66	1

Appendix I: Burst Leakage – SZ08 Results

This section shows the episodic performance of each of the optimisation algorithms as they tackle the background leakage case study on the Jowitt & Xu network. The results and discussions associated with these results were covered in section 4.2 of the thesis.

Each algorithm's step rewards, water saved%, pressure violations, and carbon emissions are listed along with the algorithm's processing speeds. A line plot of the rewards of the algorithms across the three test episodes is shown below.



NM

NM		
	Time Elapsed	
	10983	

Step	Episode 0					Episode 1					Episode 2				
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0	65.24078681	2.960504463	10.01569932	672	41.50096931	104.8251155	4.756767006	16.03019541	679	66.68135245	111.8596907	5.075982826	15.18913553	678	71.15618642
1	60.5526395	2.734123801	8.909900727	655	38.51874504	82.31494928	3.71675395	13.11512598	633	52.36218554	103.248583	4.661967016	22.09876926	670	65.67848859
2	99.69037147	4.530896377	18.85441995	668	63.4150391	43.68453114	1.985448353	8.803277885	657	27.78860395	43.66162455	1.984407256	-8.96066253	657	27.77403261
3	65.64304725	3.126435175	9.941176256	671	41.75685522	113.5625847	5.408738233	14.94976239	677	72.23943139	48.85876532	2.327036432	4.985950257	666	31.0800378
4	76.09374498	3.715319265	11.59628186	685	48.40475305	99.48272642	4.857299245	19.20569368	690	63.28295193	46.85227664	2.287588369	6.161755457	679	29.80367021
5	46.99346271	2.268887572	6.164844935	680	29.8934815	24.14653904	1.165817098	0.855896562	674	15.36009641	81.48920949	3.934373931	14.94773695	687	51.83691594
6	76.25576437	3.598894579	11.60031552	670	48.50781683	155.7882129	7.352432431	83.17472293	744	99.09999802	72.86347876	3.438795491	14.51115144	670	46.34991611
7	72.94090979	3.371126433	12.66602579	682	46.39917154	269.2970109	12.44616052	3.463777808	671	171.3052146	102.0033281	4.714310754	17.22645227	687	64.88635704
8	46.29941846	2.191374971	5.495425186	698	29.45198607	64.75608683	3.064938451	5.990560025	684	41.19264196	43.04975464	2.037566733	8.289846714	698	27.38480992
9	60.97782634	3.029212268	13.58441844	709	38.78921489	108.999527	5.414799513	15.66162288	715	69.33677913	50.65560067	2.516432221	79.06708698	616	32.2230407
10	21.78348692	0.925485252	8.986419452	715	13.8569117	509.821541	21.66009137	23.50091574	731	324.3076787	501.172445	21.29262905	23.60129401	731	318.8058157
11	42.85881573	2.178911839	9.308301659	716	27.26334986	72.57503062	3.689663159	-14.1137451	722	46.16642848	111.1322173	5.649883222	15.43602885	727	70.69342604
12	99.36524766	5.019129658	18.63363506	718	63.20822134	31.20313892	1.576130525	19.41790337	675	19.84894073	83.26426711	4.205838182	16.31318788	715	52.9660656
13	47.52865329	2.394509047	5.163934323	706	30.23392693	83.43186685	4.20332465	15.42377411	714	53.07267914	83.43087862	4.203274862	15.43881386	714	53.07205051
14	101.837966	5.056443154	18.19187122	713	64.78116693	43.22556626	2.146229223	8.115864368	702	27.49664721	321.5960084	15.96783595	4.464569979	697	204.5736528
15	378.3937657	15.50849648	12.51915652	686	240.7038423	378.4657431	15.51144648	12.51912047	686	240.7496285	341.9377276	14.01434306	20.18845743	692	217.5134273
16	304.0427534	15.35627288	-4.46570856	711	193.4076763	46.0231407	2.324488578	8.677756514	699	29.27624026	308.4491797	15.57882805	-9.10797389	716	196.2106922

17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
387.7498947	16.82823568	12.52158377	719	246.655463	417.5672685	18.12230126	17.96464245	725	265.6228908	403.3105638	17.50356431	30.96836419	737	256.5539159			
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
46.29861212	2.349282271	5.398605196	721	29.45147314	60.88149865	3.089246499	13.70840849	726	38.72793892	60.87516818	3.088925278	-13.7551365	726	38.72391198			
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
84.36568009	4.316956194	11.96200339	755	53.66669642	46.09420112	2.35862079	5.605451435	748	29.32144322	84.36299041	4.316818564	11.97219717	755	53.66498546			
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
77.62325836	3.956673192	10.58129115	749	49.37770711	108.5583802	5.533522322	17.14878836	754	69.05615684	46.26580906	2.35829686	5.646619511	743	29.43060646			
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
607.1646089	23.24080018	24.49177217	723	386.229551	602.3235957	23.05549784	25.57514932	723	383.1500857	563.7886978	21.58047468	20.12107851	717	358.6372665			
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
99.57918988	4.782052724	18.95986036	705	63.34431427	72.86330815	3.499086321	13.17552435	700	46.34980758	72.76793508	3.494506257	14.41978809	700	46.28913886			
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
108.3221628	5.388524764	18.38393718	718	65.39357572	76.83945597	3.822406243	6.780147559	691	84.67434476	46.5988999	2.318078957	5.650189252	707	68.06593553			
Total	2469.51656	4.504833435	-288.396588	16845	1567.396556	3078.136998	5.661203709	275.0148403	16820	1993.859737	2473.404724	4.394101263	240.3880725	16785	1611.805656		

PSO

PSO																		
Time Elapsed 7627																		
Step	Episode 0			Episode 0			Episode 1			Episode 1			Episode 2					
	Water	Water	Episode	Water	Water	Episode	Water	Water	Episode	Water	Water	Episode	Water	Water	Episode			
	Saved	Saved	0	Saved	Saved	0	Saved	Saved	1	Saved	Saved	1	Saved	Saved	2			
0	52.6229360	1	2.387930075	5.267630125	667	33.47450206	48.6934846	2	2.209618945	8.928699138	667	30.97489944	48.7681776	6	2.213008375	8.204555514	667	31.02241318
1	43.5870345	1.968078509	12.02566466	659	27.72658438	38.1096192	48.8397118	1.720757638	12.15234746	659	24.24229096	51.9517345	6	1.263961932	6.423780937	653	17.80688474	
2	52.2641799	2.375390721	6.047471899	657	33.24629012	8	2.219749714	7.798825486	657	31.06791752	7	2.361190178	5.478035763	657	33.0475374			

	55.5823670	-									55.6150434	-			
3	4	2.647266918	5.486961498	666	35.35705532	55.7709662	2.656249485	5.483233629	666	35.47702702	5	2.648823225	5.486903166	666	35.37784144
	55.9001249	-				55.8994329	-				51.7167955	-			
4	2	2.729354576	5.476616423	679	35.55918746	4	2.729320789	5.476672462	679	35.55874728	5	2.525101201	5.237115658	679	32.89808799
	53.4316477	-				54.5764504	-				53.9571605	-			
5	4	2.579729062	5.533134301	680	33.98893976	6	2.63500119	5.507914554	680	34.71717167	8	2.605101305	5.521657805	680	34.32322899
	53.6226664	-				53.5876882	-				49.3395795	-			
6	5	2.530724401	5.529125952	664	34.11045058	3	2.529073602	5.529818213	664	34.08820024	8	2.328583903	5.628931555	664	31.38589336
	43.8061435	-				42.9620495	-				35.5951077	-			
7	6	2.024598389	11.43205475	676	27.86596404	9	1.985586708	11.45230619	676	27.32901899	9	1.645107102	11.64198206	676	22.64275996
	42.2612953	-				33.4354249	-				46.0123953	-			
8	2	2.000248553	8.215449763	698	26.88325518	3	1.582515629	4.482127997	694	21.26894251	1	2.177790019	8.113806207	698	29.2694049
	44.9961502	-				48.5909590	-				48.5759975	-			
9	1	2.235286142	-8.54836929	704	28.62295107	3	2.41386645	8.452514124	704	30.90968086	2	2.413123203	8.452123663	704	30.90016354
	447.346695	-					-				446.579763	-			
10	3	19.00580795	12.38825262	720	284.5661798	446.914258	18.98743558	12.39751968	720	284.2910978	4	18.97322436	12.40474693	720	284.0783191
	47.1927065	-				47.4515571	-				47.6012933	-			
11	3	2.399243778	8.676476153	716	30.02022448	4	2.41240356	8.668090375	716	30.18488453	1	2.420016041	8.662581018	716	30.2801347
	48.3482362	-				48.0863379	-				48.3891648	-			
12	9	2.442162349	-8.61589007	707	30.75528007	9	2.428933363	-8.62272463	707	30.58868132	2	2.444229728	8.616576067	707	30.78131553
	47.0212015	-				48.1927656	-				36.3518576	-			
13	6	2.368943463	-9.13425094	706	29.91112674	7	2.427967245	9.104456059	706	30.6563821	7	1.83141844	9.436602466	706	23.1241437
	51.7528983	-				44.9844986	-				48.6498260	-			
14	7	2.569627016	5.525386174	702	32.92105371	6	2.233563466	8.300881381	702	28.61553929	9	2.415553744	8.203298211	702	30.94712737
	380.498716	-				378.607422	-				380.116544	-			
15	7	15.59476805	12.47073147	686	242.0428436	2	15.51725321	12.51467962	686	240.8397534	3	15.57910469	12.47948846	686	241.7997361
	46.9618100	-				48.0327681	-				45.1142553	-			
16	2	2.371897905	8.714252442	716	29.87334659	5	2.425988736	8.684581658	716	30.55460447	1	2.278583547	8.763304951	716	28.69808009
	386.927555	-				390.941416	-				393.202457	-			
17	8	16.7925464	12.54340781	719	246.1323568	8	16.96674683	12.44990062	719	248.6856541	8	17.06487537	12.40018591	719	250.1239475
	45.1966030	-				47.9617697	-				45.0357182	-			
18	9	2.293364174	9.238097265	721	28.75046316	6	2.433674147	9.166340773	721	30.50944098	8	2.285200563	9.242304381	721	28.64812111
	48.4971122	-				48.7393719	-				48.7148949	-			
19	8	2.481576739	8.005873566	748	30.84998306	6	2.493973064	7.999221091	748	31.00408929	6	2.492720586	8.002665654	748	30.98851898
	50.0424362	-				52.9201922	-				53.1021075	-			
20	3	2.550802042	5.126069148	743	31.83299453	6	2.697489264	5.057364467	743	33.6635927	3	2.706761991	5.055409381	743	33.77931264
	546.133064	-				545.043872	-				546.699456	-			
21	4	20.90465951	14.40712027	712	347.4061649	4	20.86296787	14.43124361	712	346.7133081	4	20.92633964	14.39479026	712	347.7664582

	48.8434981	-				48.4903455	-								
22	5	2.345592324	7.704301214	694	31.07032605	8	2.328633015	7.713171476	694	30.84567863	48.676324	2.337564184	7.708515876	694	30.96398322
	47.6335211	-				47.6335211	-				48.6407049	-			
23	5	2.369546561	8.620483945	707	76.44987261	5	2.369546561	8.620483945	707	75.99731475	9	2.419649281	8.594619048	707	75.44042866
Tota	2740.47060	-				2724.46588	-				2706.39932	-			
I	2	4.998714401	204.7330717	16747	1789.417396	5	4.969513169	208.9951186	16743	1778.783918	6	4.931543025	204.1539809	16741	1766.093842

DE

DE		Time Elapsed		10753											
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions
0	60.25782045	2.734386801	9.108653212	671	38.33120474	60.62594167	2.751091452	9.101781159	671	38.56537402	66.97449261	3.039176778	46.4370109	615	42.60381424
1	56.92070012	2.570131414	12.60893777	664	36.20839576	53.2947819	2.406410899	14.17587501	663	33.90187666	55.89607932	2.523866872	13.73613025	663	35.55661398
2	56.71683192	2.57776237	11.25782608	661	36.07871112	77.06002005	3.502353943	27.19489234	680	49.01941995	60.59658005	2.754095716	9.011172485	661	38.5466965
3	55.24320416	2.63111333	12.69649712	670	35.14130703	60.30208973	2.872057016	9.885082174	670	38.35936532	56.92616279	2.711268979	11.55389065	670	36.21187068
4	64.01093922	3.125369578	-9.31358658	683	40.71863866	51.86943729	2.532554019	93.58733088	578	32.99518645	53.0848494	2.591897188	93.61106479	578	33.7683344
5	26.50102763	1.279493971	133.4108527	538	16.85783369	59.70204281	2.882469499	9.892391962	684	37.97766347	56.21636157	2.714177605	12.93128434	684	35.76035192
6	64.04076934	3.022407283	9.315087943	668	40.73761419	56.73002484	2.677376334	12.81020956	668	36.0871034	56.60191643	2.671330251	12.35236665	668	36.00561108
7	33.24947696	1.536698555	28.24740739	692	21.15065728	24.58944437	1.136455881	89.13256152	575	15.64183735	24.97753896	1.154392536	89.14022866	575	15.88871208
8	32.94408493	1.559260258	85.43608333	604	20.95639131	33.12606087	1.567873271	85.43990882	604	21.07214984	56.72402844	2.684777051	-11.9526032	702	36.08328897
9	57.8127506	2.871979928	0.129942218	696	36.77584691	34.6274645	1.720198088	80.82514436	614	22.02722272	34.59183659	1.71842819	80.7904805	614	22.00455909
10	8.574861636	0.364308433	77.6329954	628	5.454640984	458.8052722	19.49263285	1.63357629	706	291.8552097	459.9297316	19.54040622	5.52525512	701	292.5705009

Step	Water			Violatio	Carbon	Water Saved			Violatio	Carbon	Water	Water Saved	Violatio	Carbon	
	Saved	Water Saved %	Rewards	ns	Emissions	Water Saved	%	Rewards	ns	Emissions	Saved	%	Rewards	ns	Emissions
0	55.4993886		70.191426					70.191426			108.25800		56.474438		
	6	2.518458097	93	584	35.30427112	55.49938866	2.518458097	93	584	35.30427112	54	4.912545108	64	605	68.86508239
1			79.169499					79.169499			78.287695		79.169499		
	78.2876957	3.534912003	45	569	49.80036899	78.2876957	3.534912003	45	569	49.80036899	7	3.534912003	45	569	49.80036899
2			93.854146					95.030215			84.031062		104.69217		
	95.9175555														
3	7	4.359423068	01	557	61.01507545	64.59878613	2.935994738	81	549	41.09257983	22	3.819185642	5	543	53.4538393
	78.1967536		127.12660					127.12660			91.717475		82.946964		
4	8	3.724340832	44	530	49.74251895	78.19675368	3.724340832	44	530	49.74251895	29	4.368303314	41	574	58.34332038
	93.5081481		108.38151					122.35005			91.185291		119.36853		
5	7	4.565587151	7	561	59.48240321	86.5761564	4.227128811	65	547	55.07282461	06	4.452172365	7	550	58.00478735
	77.9996561		127.18104					127.18104			77.999656		127.18104		
6	6	3.765895089	41	545	49.61714127	77.99965616	3.765895089	41	545	49.61714127	16	3.765895089	41	545	49.61714127
			74.919131					74.919131			110.76379		74.919131		
7	110.763791	5.227502605	22	584	70.45906272	110.763791	5.227502605	22	584	70.45906272	1	5.227502605	22	584	70.45906272
	98.7105747		61.356044					82.974235			86.211094		72.033170		
8	3	4.562128833	21	607	62.7917708	77.47499827	3.58067942	27	582	49.2833959	14	3.984437528	09	593	54.84060121
	84.8040185		79.358224					79.358224			84.804018		79.358224		
9	7	4.013817233	34	611	53.94553229	84.80401857	4.013817233	34	611	53.94553229	57	4.013817233	34	611	53.94553229
	97.2011387		62.892778					48.974591			88.697974		71.438597		
10	1	4.828687729	73	636	61.83158836	104.615812	5.197028493	05	650	66.5482103	63	4.406273706	56	624	56.42255562
	51.4609470		71.179776					71.179776			498.03335		50.019921		
11	2	2.186350958	93	635	32.73533762	51.46094702	2.186350958	93	635	32.73533762	95	21.15926301	88	657	316.8089806
	100.292308		65.913870					80.567407			84.748295		80.567407		
12	7	5.098789947	88	645	63.79794341	84.74829521	4.30854331	22	628	53.91008555	21	4.30854331	22	628	53.91008555
	97.7717398		73.489248					77.504892			95.687711		77.476332		
13	8	4.938638517	64	629	62.19455917	87.26307031	4.407825418	62	621	55.50978428	37	4.833370231	71	625	60.86886695
	86.1358254		89.778835					89.778835			99.588147		73.889779		
14	7	4.339550966	48	611	54.7927213	86.13582547	4.339550966	48	611	54.7927213	84	5.017283353	98	627	63.35001261
			87.042973					72.152292			87.396364		87.042973		
15	87.3963644	4.339390954	59	610	55.59457532	100.843306	5.007056445	7	625	64.14844378	4	4.339390954	59	610	55.59457532
	422.200436		79.903190					79.903190			439.39622		49.060794		
16	6	17.30391613	99	593	268.5701417	422.2004366	17.30391613	99	593	268.5701417	44	18.00868676	24	624	279.5087263
	84.2773661		81.370004					67.010259			102.37835		47.527462		
17	4	4.256592922	13	626	53.61051815	100.5185486	5.076885553	11	644	63.94185915	47	5.170818691	68	660	65.124919
	444.890443		52.030171					72.932593			431.11076		72.932593		
17	3	19.30811931	28	654	283.0037088	431.1107677	18.71008529	73	634	274.2381816	77	18.71008529	73	634	274.2381816

	111.483545		27.184752					68.417946			111.85772		17.110631		
18	2	5.656893461	49	689	70.91691276	87.40255002	4.434976596	17	644	55.59851012	66	5.675880157	22	699	71.15493702
	89.0690255		69.352055					41.545339			94.986613		63.427050		
19	3	4.557624392	03	671	56.65858852	109.9790731	5.627582686	38	702	69.95988795	5	4.860424868	7	680	60.42288458
			53.077885					29.336779			98.489270		47.444079		
20	-254.20171	-12.95736758	11	681	-161.7027918	86.8249686	4.425709935	88	702	55.23109903	7	5.020271827	81	689	62.65099488
	590.073279		51.012540					51.012540			198.64892		41.585580		
21	1	22.5865852	09	647	375.3574143	590.0732791	22.5865852	09	647	375.3574143	38	7.603802783	56	659	126.3645534
	88.3742675		70.280744					38.356036			101.25808		46.308159		
22	3	4.243963094	82	617	56.21663906	106.0778621	5.094135931	77	651	67.47824961	28	4.862677547	96	643	64.41229165
	71.2752845		66.087876					54.329364			99.054855		75.265712		
23	9	3.545614546	75	628	80.4326983	105.2319912	5.234803072	18	647	88.40269874	84	4.927519262	84	626	93.80681861
	2941.38784		1822.1343					1801.3022			3444.5907		1697.2402		
Total	4	5.437725644	43	14720	1906.1687	3268.687977	6.144573534	84	14735	2100.740321	63	6.540960943	63	14859	2221.96912

PPO

PPO		Time Elapsed (train)					Time Elapsed (Test)									
		3644					594									
Step	Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2	
	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	
0	55.49938866	2.518458097	70.19142693	584	35.30427112	108.8587006	4.93980353	56.65401993	605	69.24719663	113.1012413	5.132322062	40.57112148	621	71.94596159	
1	78.2876957	3.534912003	79.16949945	569	49.80036899	81.52444893	3.681060612	76.30719692	570	51.85933245	78.2876957	3.534912003	79.16949945	569	49.80036899	
2	97.87398603	4.448342224	90.42391468	559	62.25959999	99.82455512	4.536994982	41.07400556	607	63.500396	100.5923097	4.571889189	36.08836509	612	63.98878005	
3	100.0455098	4.764949435	71.1910954	586	63.64094969	87.62546096	4.173409597	123.6483975	534	55.74030823	91.6417417	4.364696288	82.9449406	574	58.29514473	
4	103.4646854	5.051720601	107.5682976	566	65.81595571	104.4350791	5.099100608	87.36807592	585	66.4332425	81.14346725	3.961874753	125.314018	544	51.61698238	
5	77.99965616	3.765895089	127.1810441	545	49.61714127	82.69357571	3.992521839	126.2227889	546	52.60303738	77.99965616	3.765895089	127.1810441	545	49.61714127	
6	110.763791	5.227502605	74.91913122	584	70.45906272	110.763791	5.227502605	74.91913122	584	70.45906272	110.763791	5.227502605	74.91913122	584	70.45906272	
7	77.47499827	3.58067942	82.97423527	582	49.2833959	98.26420283	4.541498763	61.35183976	607	62.5078247	77.47499827	3.58067942	82.97423527	582	49.2833959	

8	84.80401857	4.013817233	79.35822434	611	53.94553229	84.80401857	4.013817233	79.35822434	611	53.94553229	84.80401857	4.013817233	79.35822434	611	53.94553229
9	110.397888	5.484266278	33.08118037	666	70.22630454	86.45678404	4.294937464	75.37692592	621	54.99688947	105.7469763	5.253221656	44.99560291	654	67.26776656
10	51.46094702	2.186350958	71.17977693	635	32.73533762	51.46094702	2.186350958	71.17977693	635	32.73533762	51.46094702	2.186350958	71.17977693	635	32.73533762
11	84.74829521	4.30854331	80.56740722	628	53.91008555	84.74829521	4.30854331	80.56740722	628	53.91008555	102.2633453	5.198996051	54.93521047	656	65.05175919
12	101.1775209	5.110671066	69.54666148	633	64.36104462	87.12148959	4.400673903	86.38701046	616	55.41972196	96.85040952	4.89210035	75.4994825	627	61.60848251
13	86.13582547	4.339550966	89.77883548	611	54.7927213	110.3860589	5.561285633	50.54451395	651	70.2187798	86.13582547	4.339550966	89.77883548	611	54.7927213
14	328.5190077	16.31157566	60.10391922	634	208.9775112	98.72711992	4.901983899	74.1298894	623	62.80229552	87.3963644	4.339390954	87.04297359	610	55.59457532
15	440.1287213	18.0387082	48.07119921	625	279.9746822	422.2004366	17.30391613	79.90319099	593	268.5701417	422.4539636	17.31430695	76.92828874	597	268.7314153
16	99.40480566	5.02063379	69.9972382	641	63.23338498	84.27736614	4.256592922	81.37000413	626	53.61051815	402.8217396	20.34529844	53.09672747	656	-256.242965
17	451.9350372	19.61385269	30.11835011	676	287.4849158	448.4523477	19.46270496	43.07145094	663	285.2695074	449.685156	19.51620849	29.12767634	678	286.0537215
18	87.40255002	4.434976596	68.41794617	644	55.59851012	87.40255002	4.434976596	68.41794617	644	55.59851012	106.3865688	5.398262895	35.19391546	681	67.67462414
19	89.06902553	4.557624392	69.35205503	671	56.65858852	103.5843697	5.300368416	49.49161151	694	65.89208923	89.06902553	4.557624392	69.35205503	671	56.65858852
20	319.9346466	16.30795803	23.21290983	712	203.5168274	91.29812119	4.653718953	58.40305502	678	58.07656085	91.29812119	4.653718953	58.40305502	678	58.07656085
21	590.0732791	22.5865852	51.01254009	647	375.3574143	607.0681795	23.23710909	22.18109096	675	386.1682104	610.85357	23.38200473	11.24673926	686	388.5761729
22	100.5108285	4.826792445	46.28982672	643	63.93694826	101.5634184	4.877340559	46.30195887	643	64.60652172	88.69511872	4.259371205	68.28470846	619	56.42073892
23	71.27528459	3.545614546	66.08787675	628	67.36204221	105.242082	5.235305043	54.33261513	647	93.1620264	100.1677506	4.982880613	74.28094733	627	78.43323321
Total	2501.480084	4.514204728	1659.794592	14880	1613.263919	3428.783399	6.442563233	1668.562128	14886	2207.333129	2901.450322	5.33676164	1627.866575	14928	1860.385103

Recurrent PPO

Time Elapsed (train)				Time Elapsed (test)											
Recurrent PPO															
4519				727											
Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2	
Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	

0	55.4993886	70.1914269	6	3	584	35.30427112	55.4993886	2.518458097	3	584	35.30427112	55.4993886	70.1914269	6	3	584	35.30427112
1	78.2876957	3.534912003	5	569	49.80036899	78.2876957	3.534912003	5	569	49.80036899	78.2876957	3.534912003	5	569	49.80036899	78.2876957	3.534912003
2	64.5987861	2.935994738	1	549	41.09257983	64.5987861	2.935994738	1	549	41.09257983	64.5987861	2.935994738	1	549	41.09257983	64.5987861	2.935994738
3	78.1967536	3.724340832	4	530	49.74251895	78.1967536	3.724340832	4	530	49.74251895	78.1967536	3.724340832	4	530	49.74251895	78.1967536	3.724340832
4	104.435079	5.099100608	2	585	66.4332425	104.435079	5.099100608	2	585	66.4332425	104.435079	5.099100608	2	585	66.4332425	104.435079	5.099100608
5	77.9996561	3.765895089	1	545	49.61714127	77.9996561	3.765895089	1	545	49.61714127	77.9996561	3.765895089	1	545	49.61714127	77.9996561	3.765895089
6	110.763791	5.227502605	2	584	70.45906272	110.763791	5.227502605	2	584	70.45906272	110.763791	5.227502605	2	584	70.45906272	110.763791	5.227502605
7	77.4749982	3.58067942	7	582	49.2833959	77.4749982	3.58067942	7	582	49.2833959	77.4749982	3.58067942	7	582	49.2833959	77.4749982	3.58067942
8	62.7976263	2.972243522	4	638	39.94682607	62.7976263	2.972243522	4	638	39.94682607	62.7976263	2.972243522	4	638	39.94682607	62.7976263	2.972243522
9	86.4567840	4.294937464	2	621	54.99688947	86.4567840	4.294937464	2	621	54.99688947	86.4567840	4.294937464	2	621	54.99688947	86.4567840	4.294937464
10	51.4609470	2.186350958	3	635	32.73533762	51.4609470	2.186350958	3	635	32.73533762	51.4609470	2.186350958	3	635	32.73533762	51.4609470	2.186350958
11	78.5079714	3.991289667	7	649	49.94049077	78.5079714	3.991289667	7	649	49.94049077	78.5079714	3.991289667	7	649	49.94049077	78.5079714	3.991289667
12	87.1214895	4.400673903	6	616	55.41972196	87.1214895	4.400673903	6	616	55.41972196	87.1214895	4.400673903	6	616	55.41972196	87.1214895	4.400673903
13	108.272328	5.454795223	8	645	68.87419384	108.272328	5.454795223	8	645	68.87419384	108.272328	5.454795223	8	645	68.87419384	108.272328	5.454795223
14	87.3963644	4.339390954	9	610	55.59457532	87.3963644	4.339390954	9	610	55.59457532	87.3963644	4.339390954	9	610	55.59457532	87.3963644	4.339390954
15	433.243843	17.75653098	4	608	275.5950738	433.243843	17.75653098	4	608	275.5950738	433.243843	17.75653098	4	608	275.5950738	433.243843	17.75653098
16	84.2773661	4.256592922	3	626	53.61051815	84.2773661	4.256592922	3	626	53.61051815	84.2773661	4.256592922	3	626	53.61051815	84.2773661	4.256592922
17	431.110767	18.71008529	3	634	274.2381816	431.110767	18.71008529	3	634	274.2381816	431.110767	18.71008529	3	634	274.2381816	431.110767	18.71008529

	-														
	232.030225		69.0086378					61.4313431			87.4025500		68.4179461		
18	5	-11.77366815	9	643	-147.5990671	91.67239529	4.651636909	3	651	58.31464409	2	4.434976596	7	644	55.59851012
	89.0690255		69.3520550					55.7566297			93.0990843		56.1623374		
19	3	4.557624392	3	671	56.65858852	94.02815279	4.811380838	2	685	59.81318856	1	4.763840797	9	685	59.22218951
	91.2981211		58.4030550					58.4030550			91.2981211		58.4030550		
20	9	4.653718953	2	678	58.07656085	91.29812119	4.653718953	2	678	58.07656085	9	4.653718953	2	678	58.07656085
	590.073279		51.0125400					30.0441974			594.698187		43.0561881		
21	1	22.5865852	9	647	375.3574143	580.1797404	22.20788434	1	667	369.0639364	6	22.76361557	6	655	378.2994111
	88.3742675		70.2807448					44.4265434			88.3742675		70.2807448		
22	3	4.243963094	2	617	56.21663906	78.22312956	3.756478941	3	638	49.75929717	3	4.243963094	2	617	56.21663906
	82.2086833		67.9974689					78.3955160			105.459645		54.3361678		
23	2	4.08950038	5	630	77.9626312	89.03232683	4.428944969	9	619	86.63600835	1	5.246127799	5	647	89.04131564
Tota	2866.89478		1829.02294					1879.30668			3273.29839		1816.59756		
I	9	5.296145756	4	14696	1849.357157	3163.533177	5.937214355	9	14645	2042.387489	1	6.144758322	4	14724	2104.166899

A2C

A2C	Time Elapsed (train)					Time Elapsed (test)									
	3672					756									
Step	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
	Water Saved		Carbon			Water Saved		Carbon			Water Saved		Carbon		
	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions
0	95.40864392	4.3294652	84.90925492	577	60.69134657	55.49938866	2.518458097	70.19142693	584	35.30427112	90.06352338	4.086913662	71.62624609	586	57.29120849
1	78.2876957	3.534912003	79.16949945	569	49.80036899	78.92838002	3.563840721	77.29056734	569	50.2079211	78.2876957	3.534912003	79.16949945	569	49.80036899
2	91.91040771	4.177299444	93.60558854	554	58.46604855	64.59878613	2.935994738	95.03021581	549	41.09257983	64.59878613	2.935994738	95.03021581	549	41.09257983
3	78.19675368	3.724340832	127.1266044	530	49.74251895	103.3876554	4.924128539	98.51945227	562	65.76695534	116.7553437	5.560802379	68.96937289	591	74.27040923
4	104.4350791	5.099100608	87.36807592	585	66.4332425	104.4350791	5.099100608	87.36807592	585	66.4332425	104.4350791	5.099100608	87.36807592	585	66.4332425
5	77.99965616	3.765895089	127.1810441	545	49.61714127	86.49458846	4.176038229	123.3725051	548	55.02093761	77.99965616	3.765895089	127.1810441	545	49.61714127
6	84.25761518	3.976542324	122.9163095	533	53.59795417	110.763791	5.227502605	74.91913122	584	70.45906272	110.763791	5.227502605	74.91913122	584	70.45906272

7	77.47499827	3.58067942	82.97423527	582	49.2833959	104.0908066	4.810788222	51.95486833	617	66.21424392	80.56505756	3.723493385	82.99265828	582	51.24904441
8	62.36830134	2.951923351	40.15732256	646	39.67372385	84.80401857	4.013817233	79.35822434	611	53.94553229	84.80401857	4.013817233	79.35822434	611	53.94553229
9	86.45678404	4.294937464	75.37692592	621	54.99688947	86.45678404	4.294937464	75.37692592	621	54.99688947	104.7090555	5.201660579	47.97554824	651	66.60752439
10	503.8548009	21.40659064	40.08349654	667	320.5121159	51.46094702	2.186350958	71.17977693	635	32.73533762	51.46094702	2.186350958	71.17977693	635	32.73533762
11	84.74829521	4.30854331	80.56740722	628	53.91008555	101.8040714	5.175646892	55.92970762	655	64.75960589	84.74829521	4.30854331	80.56740722	628	53.91008555
12	100.9123071	5.097274603	69.50140203	633	64.19233677	87.12148959	4.400673903	86.38701046	616	55.41972196	87.12148959	4.400673903	86.38701046	616	55.41972196
13	86.13582547	4.339550966	89.77883548	611	54.7927213	86.13582547	4.339550966	89.77883548	611	54.7927213	86.13582547	4.339550966	89.77883548	611	54.7927213
14	87.3963644	4.339390954	87.04297359	610	55.59457532	87.3963644	4.339390954	87.04297359	610	55.59457532	87.3963644	4.339390954	87.04297359	610	55.59457532
15	422.2004366	17.30391613	79.90319099	593	268.5701417	437.713482	17.93971944	59.03626717	614	278.4383002	438.7139935	17.98072548	51.00505989	623	279.0747455
16	84.27736614	4.256592922	81.37000413	626	53.61051815	84.27736614	4.256592922	81.37000413	626	53.61051815	84.27736614	4.256592922	81.37000413	626	53.61051815
17	431.1107677	18.71008529	72.93259373	634	274.2381816	431.1107677	18.71008529	72.93259373	634	274.2381816	431.1107677	18.71008529	72.93259373	634	274.2381816
18	87.40255002	4.434976596	68.41794617	644	55.59851012	99.996075	5.073996722	38.95489941	676	63.60950323	92.80106039	4.708907587	56.43911803	656	59.03261053
19	89.06902553	4.557624392	69.35205503	671	56.65858852	92.44393011	4.730316833	58.99154287	682	58.80543282	89.06902553	4.557624392	69.35205503	671	56.65858852
20	91.29812119	4.653718953	58.40305502	678	58.07656085	91.29812119	4.653718953	58.40305502	678	58.07656085	91.29812119	4.653718953	58.40305502	678	58.07656085
21	590.0732791	22.5865852	51.01254009	647	375.3574143	590.0732791	22.5865852	51.01254009	647	375.3574143	604.2911592	23.13081144	34.14244502	663	384.4016922
22	88.37426753	4.243963094	70.28074482	617	56.21663906	88.37426753	4.243963094	70.28074482	617	56.21663906	88.37426753	4.243963094	70.28074482	617	56.21663906
23	71.27528459	3.545614546	66.08787675	628	45.33963403	71.27528459	3.545614546	66.08787675	628	45.33963403	71.27528459	3.545614546	66.08787675	628	45.33963403
Total	3654.924627	6.800813472	1905.518982	14629	2324.970653	3279.940549	6.156117214	1780.769221	14759	2086.435782	3301.055974	6.18802692	1789.558972	14749	2099.867726

DDPG

DDPG	Time Elapsed (Training)					Time Elapsed (Testing)									
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
Step	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions
			3644					807							

0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	108.3485597	4.916654293	17.48678441	678	68.9226858	108.3485597	4.916654293	17.48678441	678	68.9226858	108.3485597	4.916654293	17.48678441	678	68.9226858
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
3	108.479689	4.898166326	18.63134095	671	69.00609979	108.479689	4.898166326	18.63134095	671	69.00609979	108.479689	4.898166326	18.63134095	671	69.00609979
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
5	111.3745918	5.061940555	15.59705481	668	70.84760531	111.3745918	5.061940555	15.59705481	668	70.84760531	111.3745918	5.061940555	15.59705481	668	70.84760531
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
7	113.5643943	5.408824418	14.94971069	677	72.24058248	113.5643943	5.408824418	14.94971069	677	72.24058248	113.5643943	5.408824418	14.94971069	677	72.24058248
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
9	111.1708701	5.427979336	15.59169557	690	70.71801388	111.1708701	5.427979336	15.59169557	690	70.71801388	111.1708701	5.427979336	15.59169557	690	70.71801388
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
11	111.3192957	5.37459791	15.64106136	691	70.81243038	111.3192957	5.37459791	15.64106136	691	70.81243038	111.3192957	5.37459791	15.64106136	691	70.81243038
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
13	108.3417149	5.113192605	18.92375847	675	68.91833169	108.3417149	5.113192605	18.92375847	675	68.91833169	108.3417149	5.113192605	18.92375847	675	68.91833169
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
15	108.1649331	4.999083039	18.33191632	687	68.80587722	108.1649331	4.999083039	18.33191632	687	68.80587722	108.1649331	4.999083039	18.33191632	687	68.80587722
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
17	108.2975709	5.125778985	18.28319502	709	68.89025077	108.2975709	5.125778985	18.28319502	709	68.89025077	108.2975709	5.125778985	18.28319502	709	68.89025077
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
19	108.6305198	5.396468238	18.06190815	715	69.10204623	108.6305198	5.396468238	18.06190815	715	69.10204623	108.6305198	5.396468238	18.06190815	715	69.10204623
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
21	72.99506475	3.101241602	19.26811039	726	46.43362059	72.99506475	3.101241602	19.26811039	726	46.43362059	72.99506475	3.101241602	19.26811039	726	46.43362059
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
23	108.0903985	5.495239309	17.84587433	727	68.75846427	108.0903985	5.495239309	17.84587433	727	68.75846427	108.0903985	5.495239309	17.84587433	727	68.75846427
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
25	108.0561524	5.458123965	18.07525111	718	68.7366797	108.0561524	5.458123965	18.07525111	718	68.7366797	108.0561524	5.458123965	18.07525111	718	68.7366797
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
27	108.1431167	5.448285473	-18.0293697	717	68.7919994	108.1431167	5.448285473	-18.0293697	717	68.7919994	108.1431167	5.448285473	-18.0293697	717	68.7919994
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
29	108.2435521	5.374492336	18.22864331	713	68.85588834	108.2435521	5.374492336	18.22864331	713	68.85588834	108.2435521	5.374492336	18.22864331	713	68.85588834
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
31	443.4721377	18.17573837	22.50692441	697	282.1014962	443.4721377	18.17573837	22.50692441	697	282.1014962	443.4721377	18.17573837	22.50692441	697	282.1014962
32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
33	108.3068434	5.470248586	-18.1736145	727	68.89614921	108.3068434	5.470248586	-18.1736145	727	68.89614921	108.3068434	5.470248586	-18.1736145	727	68.89614921
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
35	453.1081926	19.66476731	22.50700737	730	288.2311835	453.1081926	19.66476731	22.50700737	730	288.2311835	453.1081926	19.66476731	22.50700737	730	288.2311835
36	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
37	108.2026407	5.490413941	19.00637803	732	68.8298638	108.2026407	5.490413941	19.00637803	732	68.8298638	108.2026407	5.490413941	19.00637803	732	68.8298638
38	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
39	108.2996197	5.541645767	18.12343237	759	68.89155406	108.2996197	5.541645767	18.12343237	759	68.89155406	108.2996197	5.541645767	18.12343237	759	68.89155406

20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
108.608873	5.53609608	17.09969754	754	69.08827631	108.608873	5.53609608	17.09969754	754	69.08827631	108.608873	5.53609608	17.09969754	754	69.08827631	
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
610.9486641	23.3856447	25.47533652	723	388.6366642	610.9486641	23.3856447	25.47533652	723	388.6366642	610.9486641	23.3856447	25.47533652	723	388.6366642	
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
108.3942322	5.205374074	17.19001391	705	68.95173898	108.3942322	5.205374074	17.19001391	705	68.95173898	108.3942322	5.205374074	17.19001391	705	68.95173898	
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
101.2457491	5.03650603	71.28938963	630	102.4779892	108.3221628	5.388524764	18.38393718	718	102.4779892	108.3221628	5.388524764	18.38393718	718	102.4779892	
Total	3753.807376	7.087770969	351.7386896	16919	2425.945491	3760.88379	7.102438416	441.4120164	17007	2425.945491	3760.88379	7.102438416	441.4120164	17007	2425.945491

SAC

SAC	Time Elapsed (training)			Time (Elapsed (testing)			Episode			Episode			Episode						
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2					
Step	Water Saved	Water Saved %	Rewards	Violatio ns	Carbon Emissions	Water Saved	%	Rewards	Violatio ns	Carbon Emissions	Water Saved	Water Saved %	Rewards	Violatio ns	Carbon Emissions				
0	108.24439	49	4.911927492	18.8490904	2	678	68.85642451	108.313824	4.915078053	18.0924634	678	68.90058974	108.244348	4	4.91192538	18.8491071	6	678	68.85639491
1	107.27751	76	4.843884869	19.8732057	8	670	68.24137451	103.2347633	4.661343018	6	670	65.66969761	107.281196	8	4.844050993	19.8670644	2	670	68.24371489
2	108.43417	04	4.92829932	17.6959334	2	668	68.97714449	108.3862433	4.926121046	4	668	68.94665708	111.714524	5	5.077390394	16.1231391	7	668	71.06384334
3	108.38171	47	5.161984694	18.7403724	7	677	68.94377637	108.4385105	5.164689753	3	677	68.97990529	111.762568	4	5.323007381	16.1193337	7	677	71.094405
4	111.54454	63	5.446224284	16.1161725	4	690	70.95571679	111.5448399	5.446238619	8	690	70.95590355	108.228522	5	5.284317586	19.0698031	2	690	68.84632775

			-					-							
	108.41825		18.2996945					18.2971503			111.734001		16.1181656		
5	38	5.23453294	3	691	68.96701959	108.4183738	5.234538734	4	691	68.96709592	7	5.394620295	9	691	71.07623313
								-							
	111.71460		-					16.1199112			111.715235		-		
6	81	5.272376464	16.1198595	675	71.06389648	111.7141821	5.27235636	9	675	71.06362551	6	5.272406083	16.1197618	675	71.0642957
								-							
	105.03187		21.4017748					14.6988511			105.031967		21.4017265		
7	92	4.854281983	4	687	66.81287898	113.549848	5.247958862	4	687	72.23132929	4	4.854286058	1	687	66.81293508
								-							
	108.30982							15.3778914			108.271977		18.6103210		
8	02	5.126358751	-18.090533	709	68.8980428	111.5349016	5.279003491	8	709	70.94958159	6	5.124567642	9	709	68.87397037
								-							
	108.28048		18.9668655					15.6364285			108.280331		18.9673243		
9	66	5.379079545	5	715	68.87938315	111.2568725	5.526938288	8	715	70.77272176	4	5.379071834	4	715	68.8792844
								-							
	506.91697		22.5029570					22.5029788			509.821435		23.5009340		
10	3	21.53668896	2	731	322.4600249	506.8322136	21.53308791	6	731	322.4061077	6	21.6600869	2	731	324.3076116
								-							
	108.05554		18.5308511					17.0901061					16.4689562		
11	36	5.493467312	2	727	68.7362924	108.1765838	5.49962091	9	727	68.81328848	121.031049	6.153132818	5	727	76.9902709
								-							
	108.05613		18.0767681					-			107.968833		18.9734777		
12	29	5.458122977	5	718	68.73666726	110.9734671	5.605483133	15.6470513	718	70.59244187	2	5.4537133	7	718	68.68113416
								-							
	121.12688		16.4704265					17.2327227			108.149558		17.8235628		
13	35	6.102411878	6	717	77.05123315	108.192024	5.450749441	5	717	68.82311034	5	5.448610014	1	717	68.79609717
								-							
	108.19388		18.7799734					18.0934350			111.312385		15.6259693		
14	61	5.372026331	6	713	68.82429485	108.2494145	5.374783417	7	713	68.85961755	4	5.526865583	3	713	70.80803463
								-							
	443.39530		22.5068378					22.5068615			429.386002		35.5093960		
15	41	18.17258934	9	697	282.0526208	443.3937254	18.17252464	6	697	282.0516166	8	17.59841709	9	709	273.1410241
								-							
	108.30726		18.1633925					12.5155453			398.803192		12.5155461		
16	96	5.470270113	3	727	68.89642033	-398.8005267	20.14219923	7	722	-253.6849911	4	20.14233387	3	722	-253.6866868

			-													
	452.96940		22.5069434					22.5069356			452.968452		22.5069234			
17	11	19.6587438	3	730	288.1428954	453.0461061	19.66207277	7	730	288.191689	3	19.65870262	5	730	288.1422919	
			-					-					-			
	108.14316		19.0697268					19.0419184			108.183212		19.0420739			
18	79	5.487396176	5	732	68.79203198	108.1832944	5.489432271	5	732	68.81755722	8	5.48942813	8	732	68.81750532	
			-					-					-			
	108.26682		18.5690392					18.5689959			111.252827		15.6506922			
19	75	5.539967804	9	759	68.87069429	108.2668684	5.539969896	9	759	68.8707203	4	5.69276016	7	759	70.77014859	
			-					-					-			
	108.61048		17.0987604					15.6133503					15.6136474			
20	03	5.53617801	8	754	69.08929876	111.5438522	5.685700124	2	754	70.95527525	111.543784	5.685696646	8	754	70.95523185	
			-					-					-			
	610.27756		25.4814509					5.37474420			607.185230		-			
21	94	23.35995681	1	723	388.2097674	610.9125809	23.38426352	6	693	388.613711	9	23.24158954	24.4917722	723	386.2426691	
			-					-					-			
	108.24942		18.8280011					15.5744381			108.394551		17.1888711			
22	89	5.198420242	9	705	68.85962672	111.5019736	5.354615933	8	705	70.92863547	4	5.205389402	7	705	68.95194202	
			-					-					-			
	108.32216		18.3839371					18.4832453			108.322543		18.3772621			
23	28	5.388524764	8	718	115.0208136	108.3187025	5.388352629	3	718	100.4885114	5	5.388543702	1	718	100.3989426	
			-					-					-			
Tot	4194.5284		459.122568					402.819099			3688.98134		454.534832			
al	23	7.872238119	1	17011	2714.33834	3695.182639	6.819696816	3	16976	2382.164399	9	6.813593569	2	17018	2378.127622	

TQC

TQC	Time Elapsed (train)					Time Elapsed (Test)									
	Episode 0	Episode 0	Episode 0	Episode 0	Episode 0	Episode 1	Episode 1	Episode 1	Episode 1	Episode 1	Episode 2	Episode 2	Episode 2	Episode 2	Episode 2
Step	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions	Water Saved	%	Rewards	Violations	Emissions
			6113					754							

0	95.03926937	4.312703676	80.87117776	581	60.45638003	94.07271217	4.268843125	85.88567828	576	59.84153367	94.03030197	4.266918629	84.87877421	577	59.81455569
1	79.71856156	3.599519664	77.29452772	569	50.71057138	107.8496693	4.869719146	38.06211071	615	68.60533166	-344.178279	15.54062766	86.90636132	564	218.9386869
2	109.3908151	4.971778525	40.96397923	607	69.58568533	96.80025554	4.399541507	93.86116435	557	61.57657855	65.89661232	2.994980535	94.03770165	550	41.91815303
3	83.00217867	3.953212744	126.1581973	531	52.79934589	100.0051539	4.763027369	71.18015132	586	63.61527847	100.5911891	4.790938948	69.19660488	588	63.98806718
4	82.46322193	4.026312507	124.321813	545	52.45650473	79.21492295	3.867712509	125.3015739	544	50.39019679	87.52411467	4.2734134	122.3571115	547	55.67583983
5	94.4418097	4.559737374	71.06989182	599	60.07632398	102.4030407	4.944112925	111.3650239	564	65.14062226	87.09955726	4.205246678	123.3775914	548	55.40577037
6	105.5297833	4.980483352	87.18467352	572	67.12960573	97.63775404	4.608018641	100.4956808	556	62.1093281	100.7419984	4.754523609	105.2785475	554	64.08400004
7	95.92806985	4.433529179	68.3363488	600	61.02176379	103.5854199	4.787430651	51.09475749	617	65.89275733	91.48741089	4.228294246	71.29478084	597	58.19697182
8	94.51712611	4.473543542	75.93528927	617	60.12423426	59.47418871	2.814943532	52.14880441	634	37.83272092	88.83227155	4.204476491	80.88771138	612	56.50798458
9	86.51150146	4.297655676	75.37743862	621	55.03169631	86.59944147	4.302024296	75.37779672	621	55.08763671	86.46317169	4.295254785	75.37688355	621	55.00095277
10	51.46094702	2.186350958	71.17977693	635	32.73533762	51.46094702	2.186350958	71.17977693	635	32.73533762	51.46094702	2.186350958	71.17977693	635	32.73533762
11	84.74829521	4.30854331	80.56740722	628	53.91008555	84.74829521	4.30854331	80.56740722	628	53.91008555	84.74829521	4.30854331	80.56740722	628	53.91008555
12	87.1620175	4.402721046	86.40132116	616	55.44550257	87.17448801	4.403350956	86.40567773	616	55.45343532	87.1712896	4.403189398	86.40445547	616	55.45140074
13	108.1162638	5.446932617	55.88431241	645	68.77491776	86.2194812	4.343765569	89.77825738	611	54.84593638	86.31879793	4.348769178	89.77920995	611	54.90911374
14	87.50933823	4.345000313	87.04090125	610	55.66644024	87.45823021	4.342462706	87.04222384	610	55.6339294	87.57224061	4.348123533	87.04229744	610	55.7064537
15	422.3018254	17.30807155	79.90389224	593	268.6346372	422.3188482	17.30876923	79.90400483	593	268.6454657	422.2881236	17.30750999	79.90379296	593	268.6259212
16	84.99743166	4.292961237	79.26298793	629	54.06856623	73.08628681	3.691365611	62.1079979	642	46.49164877	72.65951424	3.669810629	62.1048618	642	46.2201702
17	431.128655	18.7108616	72.9327137	634	274.24956	431.1735437	18.71280975	72.9330681	634	274.2781146	431.138068	18.71127012	72.93279944	634	274.2555478
18	87.40255002	4.434976596	68.41794617	644	55.59851012	87.40255002	4.434976596	68.41794617	644	55.59851012	87.40255002	4.434976596	68.41794617	644	55.59851012
19	89.06902553	4.557624392	69.35205503	671	56.65858852	89.06902553	4.557624392	69.35205503	671	56.65858852	89.06902553	4.557624392	69.35205503	671	56.65858852
20	91.29812119	4.653718953	58.40305502	678	58.07656085	91.29812119	4.653718953	58.40305502	678	58.07656085	91.29812119	4.653718953	58.40305502	678	58.07656085
21	590.0732791	22.5865852	51.01254009	647	375.3574143	590.0732791	22.5865852	51.01254009	647	375.3574143	590.0732791	22.5865852	51.01254009	647	375.3574143
22	88.37426753	4.243963094	70.28074482	617	56.21663906	88.37426753	4.243963094	70.28074482	617	56.21663906	88.37426753	4.243963094	70.28074482	617	56.21663906
23	108.256033	5.385235111	18.96994797	718	90.84401057	71.27528459	3.545614546	66.08787675	628	89.89895507	71.27528459	3.545614546	66.08787675	628	76.32539604
Total	3338.440387	6.269667592	1739.183043	14807	2145.628882	3268.775207	6.122719774	1818.245374	14724	2123.892606	2799.338153	5.240811231	1927.060887	14612	1811.700748

ARS

ARS	Time Elapsed (train)					Time Elapsed (test)										
	3222					675										
Step	Episode 0		Episode 0		Episode 0		Episode 1		Episode 1		Episode 2		Episode 2		Episode 2	
	Water Saved	Water Saved %	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	Water Saved	%	Rewards	Violations	Carbon Emissions	
0	108.348559		17.4867844	-				17.4867844	-		108.34855		17.4867844	-		
	7	4.916654293	1	678	68.9226858	108.3485597	4.916654293	1	678	68.9226858	97	4.916654293	1	678	68.9226858	
1			18.6313409	-				18.6313409	-		108.47968		18.6313409	-		
	108.479689	4.898166326	5	671	69.00609979	108.479689	4.898166326	5	671	69.00609979	9	4.898166326	5	671	69.00609979	
2			15.5970548	-				15.5970548	-		111.37459		15.5970548	-		
	111.374591	5.061940555	1	668	70.84760531	111.3745918	5.061940555	1	668	70.84760531	18	5.061940555	1	668	70.84760531	
3			14.9497106	-				14.9497106	-		113.56439		14.9497106	-		
	113.564394	5.408824418	9	677	72.24058248	113.5643943	5.408824418	9	677	72.24058248	43	5.408824418	9	677	72.24058248	
4			15.5916955	-				15.5916955	-		111.17087		15.5916955	-		
	111.170870	5.427979336	7	690	70.71801388	111.1708701	5.427979336	7	690	70.71801388	01	5.427979336	7	690	70.71801388	
5			15.6410613	-				15.6410613	-		111.31929		15.6410613	-		
	111.319295	5.37459791	6	691	70.81243038	111.3192957	5.37459791	6	691	70.81243038	57	5.37459791	6	691	70.81243038	
6			18.9237584	-				18.9237584	-		108.34171		18.9237584	-		
	108.341714	5.113192605	7	675	68.91833169	108.3417149	5.113192605	7	675	68.91833169	49	5.113192605	7	675	68.91833169	
7			18.3319163	-				18.3319163	-		108.16493		18.3319163	-		
	108.164933	4.999083039	2	687	68.80587722	108.1649331	4.999083039	2	687	68.80587722	31	4.999083039	2	687	68.80587722	
8			18.2831950	-				18.2831950	-		108.29757		18.2831950	-		
	108.297570	5.125778985	2	709	68.89025077	108.2975709	5.125778985	2	709	68.89025077	09	5.125778985	2	709	68.89025077	



			-					-							
	108.630519		18.0619081					18.0619081			108.63051		18.0619081		
9	8	5.396468238	5	715	69.10204623	108.6305198	5.396468238	5	715	69.10204623	98	5.396468238	5	715	69.10204623
			-					-					-		
	72.9950647		19.2681103					19.2681103			72.995064		19.2681103		
10	5	3.101241602	9	726	46.43362059	72.99506475	3.101241602	9	726	46.43362059	75	3.101241602	9	726	46.43362059
			-					-					-		
	108.090398		17.8458743					17.8458743			108.09039		17.8458743		
11	5	5.495239309	3	727	68.75846427	108.0903985	5.495239309	3	727	68.75846427	85	5.495239309	3	727	68.75846427
			-					-					-		
	108.056152		18.0752511					18.0752511			108.05615		18.0752511		
12	4	5.458123965	1	718	68.7366797	108.0561524	5.458123965	1	718	68.7366797	24	5.458123965	1	718	68.7366797
	108.143116		-					-			108.14311		-		
13	7	5.448285473	18.0293697	717	68.7919994	108.1431167	5.448285473	18.0293697	717	68.7919994	67	5.448285473	18.0293697	717	68.7919994
			-					-					-		
	108.243552		18.2286433					18.2286433			108.24355		18.2286433		
14	1	5.374492336	1	713	68.85588834	108.2435521	5.374492336	1	713	68.85588834	21	5.374492336	1	713	68.85588834
			-					-					-		
	443.472137		22.5069244					22.5069244			443.47213		22.5069244		
15	7	18.17573837	1	697	282.1014962	443.4721377	18.17573837	1	697	282.1014962	77	18.17573837	1	697	282.1014962
	108.306843		-					-			108.30684		-		
16	4	5.470248586	18.1736145	727	68.89614921	108.3068434	5.470248586	18.1736145	727	68.89614921	34	5.470248586	18.1736145	727	68.89614921
			-					-					-		
	453.108192		22.5070073					22.5070073			453.10819		22.5070073		
17	6	19.66476731	7	730	288.2311835	453.1081926	19.66476731	7	730	288.2311835	26	19.66476731	7	730	288.2311835
			-					-					-		
	108.202640		19.0063780					19.0063780			108.20264		19.0063780		
18	7	5.490413941	3	732	68.8298638	108.2026407	5.490413941	3	732	68.8298638	07	5.490413941	3	732	68.8298638
			-					-					-		
	108.299619		18.1234323					18.1234323			108.29961		18.1234323		
19	7	5.541645767	7	759	68.89155406	108.2996197	5.541645767	7	759	68.89155406	97	5.541645767	7	759	68.89155406
			-					-					-		
			17.0996975					17.0996975			108.60887		17.0996975		
20	108.608873	5.53609608	4	754	69.08827631	108.608873	5.53609608	4	754	69.08827631	3	5.53609608	4	754	69.08827631
			-					-					-		
	610.948664		25.4753365					25.4753365			610.94866		25.4753365		
21	1	23.3856447	2	723	388.6366642	610.9486641	23.3856447	2	723	388.6366642	41	23.3856447	2	723	388.6366642

			-					-							
	108.394232		17.1900139					17.1900139			108.39423		17.1900139		
22	2	5.205374074	1	705	68.95173898	108.3942322	5.205374074	1	705	68.95173898	22	5.205374074	1	705	68.95173898
								-							
	71.2752845		66.0878767					18.3839371			108.32216		18.3839371		
23	9	3.545614546	5	628	102.4779892	108.3221628	5.388524764	8	718	102.4779892	28	5.388524764	8	718	102.4779892
								-							
Tota	3723.8369		356.940202					441.412016			3760.8837		441.412016		
I	12	7.02565049	5	16917	2425.945491	3760.88379	7.102438416	4	17007	2425.945491	9	7.102438416	4	17007	2425.945491

Appendix J: Proof of Publications

Journal Papers

Leakage detection review paper: [Review of leakage detection in water distribution networks - IOPscience](#)

IOPscience  Journals  Books Publishing Support  Login

IOP Conference Series: Earth and Environmental Science

PAPER • OPEN ACCESS


Review of leakage detection in water distribution networks

Ahmed Negm¹, Xiandong Ma¹ and George Aggidis¹
Published under licence by IOP Publishing Ltd

[IOP Conference Series: Earth and Environmental Science, Volume 1136, 14th International Conference on Hydroinformatics 04/07/2022 - 08/07/2022 Bucharest, Romania](#)

Citation Ahmed Negm et al 2023 *IOP Conf. Ser.: Earth Environ. Sci.* **1136** 012052
DOI 10.1088/1755-1315/1136/1/012052

 Article PDF


References 

 Article and author information


Abstract

The conservation of water is, justifiably, a huge concern to water companies, regulatory bodies, environmentalist and more but, unfortunately, water loss through leakage can remain undetected for long periods of time without effective leakage detection. Leakage detection in water distribution

Article metrics
515 Total downloads



MathJax
[Turn on MathJax](#)

Share this article


[Abstract](#)

[References](#)

Challenges and Opportunities of Deep Reinforcement Learning in UWS (Accepted): [Deep reinforcement Learning Challenges and Opportunities for Urban Water Systems. - ScienceDirect](#)



Deep reinforcement Learning Challenges and Opportunities for Urban Water Systems.

Ahmed Negm, Xiandong Ma, George Aggidis [ORCID](#) [Email](#)

Show more [v](#)

[+](#) Add to Mendeley [Share](#) [Cite](#)

<https://doi.org/10.1016/j.watres.2024.121145>

[Get rights and content](#)

Under a Creative Commons [license](#)

[open access](#)

Conference Papers

Water Pressure Optimisation for Leakage Management Using Q Learning: [Water Pressure Optimisation for Leakage Management Using Q Learning | IEEE Conference Publication | IEEE Xplore](#)

Conferences > 2023 IEEE Conference on Artif...

Water Pressure Optimisation for Leakage Management Using Q Learning

Publisher: IEEE

[Cite This](#)

[PDF](#)

Ahmed Negm ; Xiandong Ma ; George Aggidis [All Authors](#)

64
Full
Text Views



Abstract

Document Sections

- I. Introduction
- II. Methodology
- III. Discussion and Results
- IV. Conclusions



Abstract:

The recent global urbanization problem has set the industry and researchers sights to the importance of safe, effective water distribution due to the unprecedented demand placed on our aging water networks. Our current water practices often increase the degradation of assets through heightened pressures causing more failures and leakage. Whilst the higher network pressures ensure customer demands are met; they cause detrimental failures to the system, long-term expenses, higher carbon emissions and energy consumption. This paper uses a baseline reinforcement learning algorithm to optimize valve set point for active pressure control. Using optimized Q-learning in an EPANET-Python environment, the agent learns to modify valve set points to decrease the average pressures whilst remaining within the OFWAT mandated pressure limits of 10m. This code is tested on the d-town test network. The agent shows continuous improvement finding an optimized set point of 26m and dropping the average system pressure by 2% by making simple changes to two pressure reducing valves. The agent learns the optimal actions to take for different states however further

Industrial Summits

12th Annual Global Leakage Summit 2022

From: Malcolm Farley <malcolm.farley@lbcg.com>
Sent: Thursday, 14 July 2022, 12:35 pm
To: Negm, Ahmed (Postgraduate Researcher) <a.negm@lancaster.ac.uk>
Subject: [External] Presenting at the 2022 Global Leakage Summit

This email originated outside the University. Check before clicking links or attachments.

Morning Ahmed

It was a pleasure to meet you at the conference last Tuesday, I hope you enjoyed being with like-minded colleagues in London and that you found the Summit agenda stimulating?!

I want to thank you, on behalf of LBCG and personally, for helping to keep up the momentum in the final afternoon of the Summit with such a motivational and thought-provoking presentation!

Your presentation was spot-on in its delivery, relating well to the final session theme on expanding the potential and maintaining the benefits of smart networks - your research on how DRL complements existing modelling systems fitted that title perfectly!

So, once again Ahmed, thank you for so successfully contributing to the Summit with such a well-presented piece of research and its future practicable applications for our industry - and for your contribution to the discussion panel that followed. Although the audience had thinned out by that time, you had a lot of interest from those who stayed!

We must certainly continue to observe and develop the DRL concept at future Summits - so please keep me in touch with any case study results?

And watch this space for the 2023 Summit date announcement!

Many thanks and best regards

Malcolm

Malcolm Farley
Conference Director
2022 Global Leakage Summit

13th Annual Global Leakage Summit 2023



London Business Conferences Group
85 Great Portland Street
London
W1W 7LT
www.lbcg.com

Ahmed Negm
Final Year PhD Candidate
Engineering Department
Lancaster University, UK

13 September 2023.

Dear Ahmed

2023 Global Leakage Summit

I am writing to thank you for giving your time to come to the Global Leakage Summit last week, and for making a great contribution to the Summit agenda.

Your presentation on the final stages of your work on how we can apply Deep Reinforced Learning to the water industry was an inspirational finale to the session on reviewing network operating practices – and to the conference itself!

I do hope that you enjoyed your Summit experience and the other sessions – I believe we had some great agenda topics covering both policy and technology, with excellent presentations from expert speakers.

Once again, thank you for supporting the Global Leakage Summit, Ahmed, and I hope our paths will cross again sometime in the future.

With my best regards

Malcolm

A handwritten signature in black ink that reads 'Malcolm Farley'.

Malcolm Farley
Conference Director
2023 Global Leakage Summit

Presentations

IMechE Webinar: 'Application of AI in leakage management in water distribution networks'

RE: [External] RE: Webinar on "Application of AI in leakage management in water distribution networks"



Evie Cleden <Evie.Cleden@IMECHE.org>

03/01/2024 15:27



To: Aggidis, George Cc: Negm, Ahmed (Postgraduate Researcher); Molly Collier

Hi George,

Yes, I can now confirm we have been able to move the date of this webinar to the 31st.

Regards,
Evie

Evie Cleden
Conference Producer

Institution Of Mechanical Engineers
1, Birdcage Walk
Westminster
London SW1H 9JJ
<http://www.imeche.org>

T 0207 227 1168

F

E Evie.Cleden@IMECHE.org