Explainable Optimisation through Online and Offline Hyper-heuristics

WILLIAM B. YATES, University of Exeter
EDWARD C. KEEDWELL, University of Exeter
AHMED KHEIRI, Lancaster University

Abstract - Research in the explainability of optimisation techniques has largely focused on metaheuristics and their movement of solutions around the search landscape. Hyper-heuristics create a different challenge for explainability as they make use of many more operators, or low-level heuristics and learning algorithms which modify their probability of selection online. This paper describes a set of methods for explaining hyper-heuristics decisions in both online and offline scenarios using selection hyper-heuristics as an example. These methods help to explain various aspects of the function of hyper-heuristics both at a particular juncture in the optimisation process and through time. Visualisations of each method acting on sequences provide an understanding of which operators are being utilised and when, and in which combinations to produce a greater understanding of the algorithm-problem nexus in hyper-heuristic search. These methods are demonstrated on a range of problems including those in operational research and water distribution network optimisation. They demonstrate the insight that can be generated from optimisation using selection hyper-heuristics, including building an understanding of heuristic usage, useful combinations of heuristics and heuristic parameterisations. Furthermore the dynamics of heuristic utility are explored throughout an optimisation run and we show that it is possible to cluster problem instances according to heuristic selection alone, providing insight into the perception of problems from a hyper-heuristic perspective.

CCS Concepts: • **Human-centered computing** → **Visualization**; • **Computing methodologies** → **Machine learning**; **Randomized search**.

Additional Key Words and Phrases: explainable artificial intelligence, hyper-heuristics, optimisation

ACM Reference Format:
William B. Yates, Edward C. Keedwell, and Ahmed Kheiri. 2024. Explainable Optimisation through Online and Offline Hyper-heuristics. 1, 1 (October 2024), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

To date, explainable AI has largely focused on the development of explainable methods in machine learning where it is used in data-driven decision support or fully automated decision making systems. Naturally, there is a desire for these decisions to be explained in a human-readable way to ensure that such decisions are free from bias and are made on rational grounds. While the evidence-basis for machine learned decisions is difficult to derive, given the complexity of methods that reside between the data and decision making layers, it is still possible in principle. Recently attention has turned to optimisation methods as an AI technique that would also benefit from generating human-readable outputs when supporting human decision making. However, there are inherent difficulties with providing explanations for

Authors' addresses: William B. Yates, University of Exeter, Computer Science, College of Engineering, Mathematics and Physical Sciences, Exeter, UK, EX4 4QF, W.B.Yates2@exeter.ac.uk; Edward C. Keedwell, University of Exeter, Computer Science, College of Engineering, Mathematics and Physical Sciences, Exeter, UK, EX4 4QF, E.C.Keedwell@exeter.ac.uk; Ahmed Kheiri, Lancaster University, LUMS Charles Carter Building, Lancaster, UK, LA1 4YX, a.kheiri@lancaster.ac.uk.

optimised solutions that have been created through the types of iterative methods that tend to prevail in metaheuristic optimisation algorithms. Explaining optimisation decision making in such algorithms is usually carried out through the analysis of the movement of solutions across the space of possible solutions (landscape analysis), the final solution sets (e.g. multi-objective surface point cloud analysis) or the hyperparameters used to calibrate the algorithm [Hinton et al. 1987; Hutter et al. 2011; Ochoa and Malan 2019]. Selection hyper-heuristics as methods that learn optimisation strategies to solve problems, possess elements of machine learning and optimisation and therefore present a unique opportunity to explain decision making and optimisation strategies across domains. Most obviously, selection hyper-heuristics allow for a greater understanding of the usefulness of heuristics in solving the problem at hand than metaheuristics which do not typically adapt to problem types. These methods usually require a range of heuristic types and parameterisations to function effectively and through an understanding of their learned probability of selection, a greater understanding of the type of landscape being traversed. A finer-grained understanding can be achieved through the plotting of changes to these probabilities through time which can reveal the effects of landscape change as optimality increases, providing that the hyper-heuristic is suitably dynamic in its decisions. It is also possible to make use of work in the explainable machine learning field to better understand the decision making that has led to the probability of heuristic selection described earlier. These methods will be specific to each problem, but offer an insight into the learning processes during the online (or occasionally offline) processes involved. Here, we explore the use of traditional methods of explaining machine learning decisions alongside bespoke approaches to explain the decisions made by selection hyper-heuristics.

Furthermore, recent methods provide an additional opportunity to explore the interaction between heuristic selections as part of a sequence of operations. The understanding of heuristic sequences is critical in optimisation as the ordering of diversification (exploration) and intensification (exploitation) operators is crucial. Diversifying operations attempt to find new, previously unvisited areas of the search landscape often through the use of randomness whereas intensification operators generate new solutions in promising areas that rely on previous knowledge built up by the optimiser. In practice, however, many operators exist on the continuum between diversification and intensification. Intensification operators are useful as a standalone operator or after a diversification operation to find local optima within the search space. Diversification tends to be more useful as a pre-operation to intensification. A simple example of this can be seen in the case of a random perturbation operator, A and a deterministic local search operation, B. In this case, the ordering B-A makes little sense as the local search operation will intensively search the local landscape only for the random perturbation to move the solution away from the local area. The reverse ordering, A-B though, plays to the strengths of both operations and can be considered to be much the more logical ordering. However, in reality not every operator can be described as purely diversifying or intensifying, particularly when operator parametrisations are also considered and so automated methods for evaluating and creating these sequences of operation is required. Together the potential for understanding heuristic usage, machine learning decision-making and heuristic-heuristic interactions allows for a different form of algorithmic understanding from other approaches. This necessitates a new set of tools to provide explanation of the simultaneous learning and optimisation seen in selection hyper-heuristics. Furthermore, through experimentation on a variety of domains and problem instances, it is also possible to determine mappings between these algorithmic insights and problems that leads to an illumination of the *algorithm-problem nexus*. This greater understanding of algorithm decisions, mapped to problem types raises the potential for the informed modification of algorithms to achieve better and more feasible solutions. This is especially the case for hyper-heuristics where significant elements of the algorithm are manually determined including the utility, variety and cardinality of the set of low-level heuristics. An understanding of algorithm behaviour can therefore be used to tune these aspects for a given domain and to yield the best possible performance.

In this paper we present several methods for explaining the decisions made by selection hyper-heuristics and explore the potential for strategies to be modified according to problem domain and state of the optimisation. Section 2 examines previous literature and research within this area. Section 3 briefly describes the underpinning mathematical methods used in the later sections and the algorithmic method (SSHH) used within the online experimentation sections. Section 4 describes a set of methods to visualise the online learning process within a selection hyper-heuristic. Section 5 describes a set of techniques to explain offline learning in selection hyper-heuristics with a particular focus on capturing the changes seen in heuristic utility over time. Section 6 describes methods to elucidate the mapping between sequences and problem spaces, and present examples of visualisation of the probability matrices within SSHH, allowing for an understanding of heuristic usage, the creation of sequences and the similarities and differences between strategies for problem solving in each domain. Together these methods seek to uncover information about the configuration of the learned probability profiles for a set of heuristics on a set of operational research problems and collectively provide a set of methods that move towards explainable hyper-heuristics. Finally, Section 7 concludes the paper.

2 RELATED WORK

Hyper-heuristics can be defined as heuristic methods that search within the domain of heuristics rather than solutions. This approach has gained popularity in addressing NP-hard optimisation problems because it generalises well across problem domains. More comprehensive literature on hyper-heuristics can be found in the following review papers: [Burke et al. 2013, 2019; Drake et al. 2020; Sánchez et al. 2020].

Hyper-heuristics can be classified into two main types: *selection hyper-heuristics*, which involve selecting a low-level heuristic (LLH) or a sequence of LLHs, and *generation hyper-heuristics*, which focus on generating heuristics. A variety of metric methods have emerged in the literature on hyper-heuristics to evaluate the performance of LLHs or heuristic sequences in studied problems. These methods typically quantify performance metrics and subsequently rank LLHs accordingly, or normalise the metrics to determine selection probabilities. This approach allows for the visualisation and inspection of either the rank of LLHs or the selection probability in each iteration, providing explainability to users. However, the extent of explainability depends on the metric mechanisms, which we categorise into three groups: function-based, reward, and statistical mechanisms.

Table 1 illustrates the metric mechanisms, application domains of explainable hyper-heuristics, and the types of optimisation problems addressed.

2.1 Function-based Mechanism

The function-based mechanism is commonly employed as a white-box metric method for hyper-heuristics. Specifically designed functions, comprising variable performance indicators, are applied to hyper-heuristics. Each low-level heuristic (LLH) is assigned a value during the iteration based on its previous performance, and LLHs are then selected either by their rank or probability normalised according to these respective values. The visual representation of both the rank and probability in each iteration provides a clear means for users to examine and understand the process, enhancing explainability. However, since all functions are provided by the designers, it is crucial to offer appropriate explanations regarding why these functions are designed in a particular way and how the parameters were determined. This additional information significantly contributes to enhancing explainability by providing insights into the rationale behind the design choices and parameter selections.

Kendall et al. [2002] introduced a choice-function-based hyper-heuristic, which ranks and selects the best low-level heuristic by considering recent improvements of each LLH, recent improvements for consecutive pairs of LLHs, and

Table 1. Metric mechanisms, application domains of explainable hyper-heuristics, and addressed optimisation problem types. Note that HyFlex (Hyper-heuristics Flexible framework) is a software tool, developed for designing and comparing the performance of selection hyper-heuristics [Ochoa et al. 2012].

Metric Mechanism	References
Function-based	Kendall et al. [2002]; Kheiri [2014]; Kheiri and Özcan [2016]; Maashi et al. [2015, 2014]; Misir et al. [2011]; Mısır et al. [2012]; Özcan and Kheiri [2012]; Qu et al. [2015]
Reward	Chen et al. [2024]; Hsiao et al. [2012]; Kheiri et al. [2021]; Kheiri and Keedwell [2015]; Soria-Alcaraz et al. [2014]; Zhang et al. [2020, 2022]; Zhao et al. [2021a,b]
Statistical	Qu et al. [2009]; Soria-Alcaraz et al. [2017]
Application Domain	**References**
HyFlex	Hsiao et al. [2012]; Kheiri et al. [2021]; Kheiri and Keedwell [2015]; Misir et al. [2011]; Mısır et al. [2012]; Zhao et al. [2021a]
Bin Packing	Zhang et al. [2022]
Design Problems	Maashi et al. [2015, 2014]
Vehicle Routing	Soria-Alcaraz et al. [2017]; Zhang et al. [2022]; Zhao et al. [2021b]
Scheduling Problem	Chen et al. [2024]; Cowling et al. [2001]; Kendall et al. [2002]
Timetabling	Qu et al. [2009, 2015]; Soria-Alcaraz et al. [2017, 2014]
Optimisation Type	**References**
Single-objective	Chen et al. [2024]; Cowling et al. [2001]; Hsiao et al. [2012]; Kendall et al. [2002]; Kheiri et al. [2021]; Kheiri and Keedwell [2015]; Misir et al. [2011]; Mısır et al. [2012]; Soria-Alcaraz et al. [2017]; Zhang et al. [2022]; Zhao et al. [2021a,b]
Multi-objective	Maashi et al. [2015, 2014]; Zhang et al. [2020]

the elapsed time since the heuristic was last called. The choice-function-based performance metric enables users to inspect the ranks of LLHs in each iteration for the purpose of explainability. Maashi et al. [2015, 2014] adopted the choice-function concept from [Kendall et al. 2002] and extended its application to multi-objective optimisation problems. In [Maashi et al. 2014], a multi-objective selection hyper-heuristic (HH-CF) was introduced, utilising the choice-function as its learning mechanism. This approach employs adaptive learning and management of three low-level heuristics (NSGA-II, SPEA2, and MOGA). The choice-function integrates four performance evaluation metrics to effectively rank the performance of each low-level heuristic. The top-ranked low-level heuristic is selected and implemented in the subsequent stage. Maashi et al. [2015] expanded upon HH-CF to explore the efficacy of employing various combinations of learning and move acceptance strategies.

Another paper by Qu et al. [2015], addressing timetabling using graph colouring, proposes a hybrid hyper-heuristic approach based on estimation distribution functions. This evolutionary algorithm enables old heuristic sequences to be continuously replaced by new ones. The generation of new heuristic sequences is determined by the probabilities of each LLH appearing at each stage of sequences selected by tournament selection.

Mısır et al. [2012] implemented an adaptive selection hyper-heuristic (AdapHH), which is first introduced in [Misir et al. 2011], on HyFlex. In the 2011 international cross-domain heuristic search challenge (CHeSC 2011), AdapHH outperformed 19 selection hyper-heuristic algorithms developed by other competitors, emerging as the winner. During the LLHs search process, AdapHH provides an active heuristics list, along with a set of selection probabilities, demonstrating its explainability. The active heuristics list is generated by excluding poor-performing heuristics in each phase through a performance metric function based on simple quality indicators, such as improvement capability and speed.

After evaluation, all heuristics are ranked, and those performing below average are excluded for a certain number of phases determined by the tabu duration. The selection probability dynamically calculates the choice of heuristics from the active list. The primary finding indicates that employing online and adaptive methods in the sub-mechanisms of a hyper-heuristic can be beneficial, provided these methods are appropriately designed to accommodate evolving search environments. The study also explored various other aspects, including changes in the heuristic set size throughout the search process, the effectiveness of combining existing heuristics to explore new sets of heuristics, the significance of implementing adaptive move acceptance methods, and the impact of re-initialisation frequency on finding high-quality solutions. Insights and further elaboration on the mentioned findings can be found in [Misir 2012].

Kheiri and Özcan [2016] introduced multi-stage hyper-heuristics as a variant of selection hyper-heuristics. In this approach, the problem-solving process is divided into multiple stages. The heuristics used in each stage are dynamically selected based on the characteristics of the problem instance being solved. A function was implemented to determine a trade-off between quality and time is considered in determining good quality low-level heuristics, which was visualised for explainability in [Özcan and Kheiri 2012].

2.2 Reward Mechanism

The reward mechanism represents another popular method for selecting low-level heuristics in hyper-heuristics. LLHs are given scores if they improve the current solution; otherwise, scores are reduced or maintained if they make it worse. The reward for each LLH can be ranked or normalised as selection probability, which can be visualised and recorded during the running process, making it explainable to users.

Hsiao et al. [2012] proposed a hyper-heuristic based on Variable Neighborhood Search (VNS), comprising perturbation and local search. This approach achieved second place in CHeSC 2011. Explainability is reflected in the local search through the utilisation of adaptive ranking values to aid in selecting low-level heuristics in each iteration. During each iteration, the algorithm randomly selects one heuristic among those with the highest reward value, and the perturbation part is triggered when all LLHs are applied consecutively without improvement. All reward values are set to 1 if the selected LLH improves the solution. If the selected LLH worsens the solution or causes a draw, the reward value of the LLH is set to -1 or 0, respectively. In each iteration, both the ranking values and the selected LLH are observable, providing users with an explainable process.

Kheiri and Keedwell [2015] introduced a Sequence-based Selection Hyper-heuristic (SSHH) based on the hidden Markov model (HMM). SSHH constructs a reward-based probabilistic model of heuristic usage and transitions between heuristics. The reward for each low-level heuristic is converted into transition probabilities, allowing the learned approach to generate sequences of heuristics online during optimisation. These probabilities are visualised for explainability.

Both Soria-Alcaraz et al. [2014] and Zhang et al. [2020] developed explainable hyper-heuristics integrated with an adaptive strategy. This strategy determines how to reward a low-level heuristic based on its historical performance in search processes and automatically selects and applies an LLH in the next step based on the current reward values of LLHs. The adaptive strategy offers an explainable search process through the selection probability of each LLH normalised by its reward.

2.2.1 Reinforcement Learning. Reinforcement learning represents an interdisciplinary area of machine learning and optimal control focused on determining how an intelligent agent should act within a dynamic environment to maximise cumulative reward. This technique has motivated researchers in the hyper-heuristic community to apply it in evaluating

and rewarding low-level heuristics. However, in certain studies, reinforcement learning is combined with neural networks, which are described as black-box models. Consequently, while the rewards provided by reinforcement learning can be visualised, the integration with neural networks may pose challenges in supporting explainability.

Zhao et al. [2021a] introduced a Cooperative Multi-Stage Hyper-heuristic (CMS-HH) implemented across the six problem domains in HyFlex. CMS-HH selects low-level heuristics based on multi-armed bandits (MAB) and relay hybridisation technology (RH), incorporating ideas from both reinforcement learning and reward mechanisms. MAB measures the accumulated reward for each LLH and selects the one that maximises the reward. RH calculates and updates the selection probability of all LLHs using the reward mechanism during the search process and chooses LLHs using the roulette wheel strategy. Both methods provide traceable and transparent reward or probability metrics during the search process, offering explainability to users.

In [Zhao et al. 2021b], the location-routing problem with simultaneous pickup and delivery is addressed. An iterated local search (ILS) based hyper-heuristic approach is introduced. The authors explored the performance of the hyper-heuristic framework by combining four selection mechanisms and five acceptance strategies. The most successful approach, called fitness ratio rank based on multi-armed bandit with tabu search, which adaptively selects appropriate heuristics based on the recent performance of each LLH. In contrast to [Zhao et al. 2021a], in this approach, the reward value of each low-level heuristic is normalised as the selection probability. Tabu search is employed to prevent the algorithm from repeatedly selecting LLHs that do not make any improvement.

In [Zhang et al. 2022], a deep reinforcement learning-based hyper-heuristic is proposed, enhancing existing hyper-heuristics with a data-driven module. The deep reinforcement learning agent selects and executes an action (low-level heuristic) based on the state of the partial solution and receives a reward from the problem model. This reward is based on the Q-function, defined as the expectation of cumulative reward. A deep Q-network is utilised to approximate the Q-function. Similarly, Chen et al. [2024] introduced two reinforcement learning-based hyper-heuristics called reinforcement learning-assisted genetic programming hyper-heuristic (DRL-GPHH) and reinforcement learning-assisted genetic programming ensemble hyper-heuristics (DRL-GPEHH) methods. These approaches use a reinforcement learning agent to select LLHs generated by genetic programming (GP). However, the metric methods in [Zhang et al. 2022] and [Chen et al. 2024] are combined with neural networks, which are black-box models. Therefore, their explainability is limited.

2.3 Statistical Mechanism

The statistical mechanism utilises statistical insights to explore effective low-level heuristics or sequence patterns. Qu et al. [2009] introduced a random iterative graph-based hyper-heuristic (GHH) for timetabling problems. This approach generates a collection of heuristic sequences by randomly changing n heuristics in the initial sequences in each iteration. To evaluate the performance of a sequence, the algorithm calculates the hybridisation rates of LLHs at each position in the heuristic sequences based on frequency. By observing the sequences corresponding to good solutions, the authors then develop an iterative hybrid approach to adaptively hybridise LLHs at different stages of solution construction. The frequencies of LLHs at each position in the sequence corresponding to good solutions can be visualised to provide explainability.

Soria-Alcaraz et al. [2017] focused on identifying an effective heuristic pool for selection operations. They proposed two fitness landscape probing techniques to evaluate the performance of low-level heuristics. The first metric generates 100 representative neighbours of a randomly generated initial solution using a selected LLH and records the frequency of improved or equal solutions as its evolvability. This procedure is repeated 500 times, and each LLH is assigned an average evolvability, which determines whether to retain the LLHs in the pool. The second method is similar but

applies hill-climbing using the selected LLH on a randomly generated initial solution for 100 iterations and records the difference between the initial and final solutions to measure their quality. The improvement frequencies or improvement gap of every LLH can be visually inspected, providing understandable explanations to users.

2.4 Other Initiatives

The recent paper by van Stein et al. [2024] introduces an approach called Explainable Benchmarking, which introduces the IOHxplainer software framework. This framework enables the examination of the effects of various algorithmic components and configurations, providing insights into their performance across a wide range of scenarios. The framework operates with a large collection of modular algorithms, analysing the performance of these algorithms with diverse configurations by making option choices per module in a fully independent manner. It automatically extracts meaningful visualisations and statistics from extensive empirical experiments, facilitating the benchmarking of thousands, or even millions, of algorithm configurations. This paper serves as an example of applying explainable artificial intelligence (xAI) to explain the performance of iterative optimisation heuristics. The idea could potentially be extended to the selection of low-level heuristics in hyper-heuristics, where the LLH sequences can be considered as a collection of modular algorithms (each/several LLHs is a module). xAI techniques can assist hyper-heuristics in evaluating the performance and then provide effective and explainable choices.

Search landscape analysis, also known as fitness landscape analysis, is a valuable technique used in various fields such as optimisation and evolutionary computation. It involves studying the structure and properties of the problem space in order to gain insights into the behaviour and performance of search algorithms. By analysing the search landscape, researchers aim to understand the distribution and characteristics of different solutions, identify the presence of peaks or valleys representing optimal or suboptimal solutions, and explore the connectivity and ruggedness of the problem space. This analysis can provide valuable information on the difficulty of the problem, the effectiveness of search algorithms, and the potential for improvement. One common approach in search landscape analysis is to visualise and analyse the fitness landscape using, for example, contour plots, heatmaps, or three-dimensional surface plots. These visualisations allow researchers to observe the distribution and concentration of high-quality solutions, identify potential plateaus, and explore the presence of multiple optima or deceptive regions. The study in [Pitzer and Affenzeller 2012] presents a comprehensive survey on fitness landscape analysis, offering insights and methodologies for conducting rigorous analysis of fitness landscapes. Incorporating search landscape analysis into research can help researchers gain a deeper understanding of problem spaces, improve algorithm design, and guide the development of more efficient and effective optimisation techniques [Ochoa and Malan 2019].

Automated Machine Learning (AutoML) aims to automate the process of developing machine learning models by automatically selecting algorithms, hyperparameters, and feature engineering techniques. However, the lack of transparency and interpretability in AutoML models can make it challenging to trust their results. Explainable AutoML systems aim to enhance transparency and interpretability in the automated machine learning process. Several initiatives have been undertaken in this area to address the need for understanding and trust in AutoML systems. The work in [Zöller et al. 2022] introduces XAutoML, a visual analytics tool that enhances trust in AutoML systems. XAutoML provides visualisations and interactive features to enable users to explore and understand the model's performance, configuration choices, fairness, and robustness. The study in [Moosbauer et al. 2021] focuses on explaining the process of hyperparameter optimisation using Partial Dependence Plots (PDPs). Hyperparameter optimisation plays a crucial role in machine learning model performance. The authors propose the use of PDPs, which are graphical visualisations, to gain insights into how hyperparameters impact model performance. The paper presents a detailed explanation

of the methodology and demonstrates its effectiveness through experiments on various datasets and models. By providing interpretable visualisations, the approach enhances the understanding of hyperparameter optimisation and aids in making informed decisions during the model development process. InterpretML [Nori et al. 2019] is an open-source Python package that provides a unified framework for interpretability in machine learning. It offers various interpretability techniques, and allows practitioners to easily compare interpretability algorithms by exposing multiple methods under a unified API. Similar to these initiatives, in this study we aim to improve the explainability of online and offline hyper-heuristics, enabling users to understand and trust the heuristic selection learning process, enhance transparency, and facilitate decision-making.

3 PRELIMINARIES

3.1 The SSHH Hyper-heuristic

In Section 4 the SSHH hyper-heuristic (Sequence-based Selection Hyper-Heuristic) is used as the basis for experimentation. This online algorithm has been selected as it explicitly discovers heuristic sequences through its probability matrices and makes these available for inspection to provide explainability, and has been widely applied across domains. [Kheiri and Keedwell 2015, 2017; Yates and Keedwell 2021].

The SSHH hyper-heuristic uses a hidden Markov model (HMM) [Rabiner 1989] to create a probabilistic model of heuristic usage and parameterisations, transitions between heuristics and the termination of a sequence through objective function evaluation and acceptance. This enables the learned approach to generate sequences of heuristic selections, their parameters, and acceptance check decisions online during an optimisation. The HMM consists of a finite set of hidden states, and four probability matrices: a state transition matrix to determine the probability of moving from one hidden state to another, a heuristic emission matrix to determine which low-level heuristic to apply, a parameter emission matrix to determine the parameter for a heuristic, and an acceptance strategy emission matrix to determine whether a solution should be checked for acceptance or not. Two acceptance strategies are used, one which continues to build the sequence by adding further heuristics and another which stops the heuristic sequence, calculates the objective function and determines whether to accept the solution according to a given threshold. Using this method, sequences of arbitrary length can be generated as the determinant of a sequence terminating is provided by the HMM acceptance strategy probabilities. A flowchart illustrating the operation of the SSHH optimiser, and its interactions with an arbitrary problem is shown in Figure 1.

In the absence of *a priori* knowledge regarding a given problem, the number of hidden states is set to be the number of low-level heuristics in the domain, and the state transition, parameter, and acceptance matrices are set to be *equiprobable*. Equiprobability is a property of a collection of events such that each event has the same probability of occurring. In this context it refers to a general state of the HMM in which the probabilities of all subsequent hidden states are identical. The low-level heuristic emission matrix is set to the identity matrix. This ensures that, initially, each equiprobable hidden state emits a single low-level heuristic together with an equiprobable choice of heuristic parameter and acceptance check decision.

The SSHH hyper-heuristic employs an online learning algorithm. During optimisation, SSHH keeps a history of the heuristic selections, parameters, and acceptance checks produced by the HMM. If, following an acceptance check, a new, dominant (or "best") solution is found, the online learning algorithm steps through the history, increasing the probabilities of the accepted state transitions and emissions that led to the new minima. Thus the probability that the

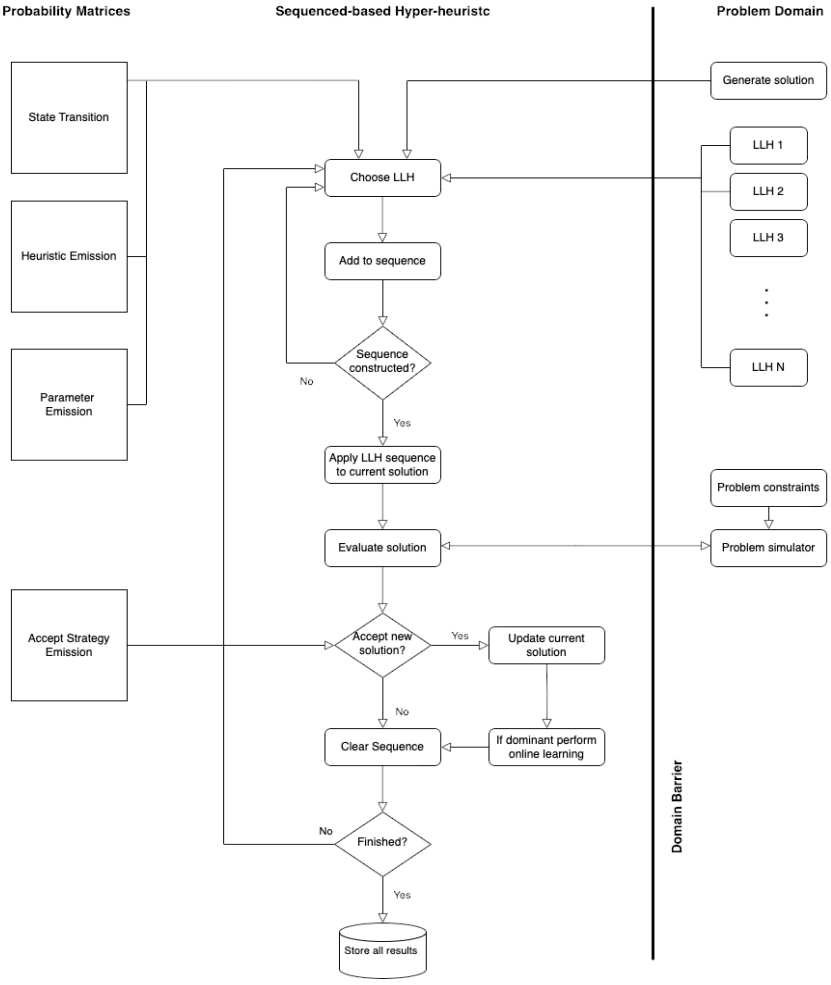


Fig. 1. A flowchart illustrating the operation of SSHH, and its interactions with the low-level heuristics of an arbitrary problem.

HMM produces the sequence of heuristic selections and parameter emissions contained in the history is now higher. After the acceptance check, the history is erased and the optimisation process is resumed.

3.2 The Baum-Welch Learning Algorithm

The SSHH hyper-heuristic can be also be trained offline on sequences of heuristic selections using the Baum-Welch learning algorithm [Rabiner 1989]. Adopting the notation of [Rabiner 1989] (Section III-C), consider a HMM consisting of N states $S_1, \ldots, S_N$, and a sequence of *training observation* $O = O_1, O_2, \ldots, O_T$. Let the model's parameterisation λ be specified by a probability transition matrix A, a probability emission matrix B, and an initial state distribution π. The Baum-Welch learning algorithm estimates a HMM's parameters $\lambda = (A, B, \pi)$ so that they maximise (locally) the probability $P(O|\lambda)$ of the model producing an observation sequence O.

In what follows, the value $a_{ij} \in A$ is the probability that the model passes from state S_i to state S_j, the value $b_j(O_t) \in B$ is the probability that the model emits the observation O_t while in state S_j at time t, and the value π_i is the probability that the model begins its computation in state S_i.

The probability $P(O|\lambda)$ can be calculated efficiently using the *forward-backward* procedure. The *forward variable*[1] $\alpha_t(i)$ is defined recursively for each state by

$$\begin{aligned} \alpha_1(i) &= \pi_i \, b_i(O_1) \\ (\forall t = 1, 2, \ldots, T-1) \quad \alpha_{t+1}(j) &= \left[\sum_i^N \alpha_t(i) \, a_{ij} \right] b_j(O_{t+1}). \end{aligned}$$

Similarly, the *backward variable* $\beta_t(i)$ is also defined recursively for each state by

$$\begin{aligned} \beta_T(i) &= 1 \\ (\forall t = T-1, T-2, \ldots, 1) \quad \beta_t(i) &= \sum_j^N a_{ij} \, b_j(O_{t+1}) \, \beta_{t+1}(j). \end{aligned}$$

The function $\xi_t(i, j)$ is the probability of being in state S_i at time t, and state S_j at time $t+1$, given the model parameters λ and the observation sequence O, and is defined by

$$(\forall t = 1, 2, \ldots, T-1) \quad \xi_t(i, j) = \frac{\alpha_t(i) \, a_{ij} \, b_j(O_{t+1}) \, \beta_{t+1}(j)}{\sum_i^N \sum_j^N \alpha_t(i) \, a_{ij} \, b_j(O_{t+1}) \, \beta_{t+1}(j)}.$$

The function $\gamma_t(i)$ is the probability of being in state S_i at time t given the model parameters λ and the observation sequence O, and is defined by

$$(\forall t = 1, 2, \ldots, T) \quad \gamma_t(i) = \frac{\alpha_t(i) \, \beta_t(i)}{\sum_j^N \alpha_t(j) \, \beta_t(j)}.$$

Thus, the Baum-Welch algorithm is specified by the three update equations:

$$\bar{\pi}_i = \gamma_1(i) \qquad \bar{a}_{ij} = \frac{\sum_t^{T-1} \xi_t(i, j)}{\sum_t^{T-1} \gamma_t(i)} \qquad \text{and} \qquad \bar{b}_j(k) = \frac{\sum_t^T \delta(O_t, v_k) \gamma_t(j)}{\sum_t^T \gamma_t(j)}$$

where δ is the Kronecker delta function such that $\delta(O_t, v_k) = 1$ if $O_t = v_k$ or 0 otherwise. Here, the term $\delta(O_t, v_k)\gamma_t(j)$ is the probability of being in state S_j while observing the symbol $O_t = v_k$.

Applying these update equations iteratively to an initial model λ increases the probability of O being produced by the model. In practice, when implementing the Baum-Welch algorithm the update equations are modified slightly to prevent underflow issues when multiplying probabilities, and to deal with multiple sequences of observations ([Rabiner 1989], Section V-A and Section V-B).

The output from the training process of SSHH is a set of probability matrices that describe the heuristic usage and parameterisations (optional), transitions between heuristics and the probability of ending the sequence and calculating the objective function. These matrices can be used as input data to various statistical tools to explain the behaviour of the model. Here we investigate the use of the Kullback-Leibler Distance method to provide a metric of learning during optimisation by comparing the matrices of HMMs during the optimisation process.

[1] It should be noted that $P(O|\lambda) = \sum_i^N \alpha_T(i)$.

3.3 Probability Vectors

A *probability vector* is any vector with non-negative components that sum to 1. The HMM's initial state vector π, and the state transition and emission vectors $a_i \in A$ and $b_i \in B$ defined in Section 3.2 are probability vectors. The mean of a probability vector with n components is $1/n$. The probabilistic length of such a vector is

$$\sqrt{n\sigma^2 + \frac{1}{n}}$$

where σ^2 is the variance of the elements of the probability vector.

The length of a probability vector measures uncertainty; the shortest vector corresponds to maximum uncertainty, while the longest vector corresponds to minimum uncertainty (or equivalently maximum certainty). The shortest probability vector has the value $1/n$ for each component, and has a length of $1/\sqrt{n}$. The longest probability vector has the value 1 for a single component, and 0 for all the others, and has a length of 1.

3.4 The Kullback-Leibler Distance

Section 4 describes methods of visualising distances between the learned HMMs over time using the Kullback-Leibler distance. Mathematically, a HMM encodes a stationary probability distribution defined over the emission symbols. The non-symmetric Kullback-Leibler divergence D is a measure of how one probability distribution differs from another, and it can be used to define a "distance" between HMMs [Juang and Rabiner 1985]. The Kullback-Leibler distance was chosen because it is a well known and widely used measure in probability theory and statistics. Furthermore it has a clear mathematical interpretation in terms of information theory.

The symmetric Kullback-Leibler distance $D_{\mathrm{KL}}(\lambda_0, \lambda)$ between two HMMs λ_0 and λ is defined by

$$D_{\mathrm{KL}}(\lambda_0, \lambda) = \frac{1}{2}\big[D(\lambda_0, \lambda) + D(\lambda, \lambda_0)\big]$$

where

$$D(\lambda_0, \lambda) = \lim_{T\to\infty} \frac{1}{T}\big[\log P(O_T|\lambda_0) - \log P(O_T|\lambda)\big].$$

It should be noted that the Kullback-Leibler distance D_{KL} is not a metric in the technical sense as it does not satisfy the triangle inequality [Lu et al. 2013].

In this study, the distance D_{KL} is estimated over a number $i = 1, \ldots, S$ of Monte Carlo trials. Specifically, in order to estimate $D(\lambda_0, \lambda)$, the first HMM λ_0 is used to generate S sequences of length T. For each such sequence O_T^i, the probabilities that the HMMs λ_0 and λ generate this sequence, denoted $P(O_T^i|\lambda_0)$ and $P(O_T^i|\lambda)$ respectively, are calculated using the forward-backward algorithm described in Section 3.2. As the sequences were generated by λ_0, the difference $P(O_T^i|\lambda_0) - P(O_T^i|\lambda)$ is non-negative. When S and T are large

$$D(\lambda_0, \lambda) \approx \frac{1}{TS}\sum_{i=1}^{S}\big[\log P(O_T^i|\lambda_0) - \log P(O_T^i|\lambda)\big].$$

This process is then repeated for the second HMM λ in order to estimate $D(\lambda, \lambda_0)$ and thus calculate the KL-distance D_{KL}.

3.5 Logarithmic Returns

In Sections 5 and 6, logarithmic returns are used as a method to provide an assessment heuristic and heuristic subsequence effectiveness, independent of absolute objective function values and is explained further here Each problem

domain has its own objective function f, and the range of f may differ between problem instances, and problem domains. Without *a priori* knowledge of the objective functions, the objective function values from different problems or problem domains cannot be compared directly. Instead, following the methodology introduced in [Yates and Keedwell 2019], normalised subsequences of objective function values are compared.

Consider a series of objective function values $o_0, o_1, \ldots, o_n$ observed after applying a subsequence s of n low-level heuristics to some initial solution x_0. The *log returns* of this series are simply the sequential differences of the log objective values[2]

$$\log(o_1) - \log(o_0),\ \log(o_2) - \log(o_1), \ldots, \log(o_n) - \log(o_{n-1})$$

and such subsequences are invariant to scaling. The sum of the n log returns is equal to the log return over the whole subsequence. In symbols

$$\sum_{i=1}^{n} \big(\log(o_i) - \log(o_{i-1})\big) = \log\left(\frac{o_n}{o_0}\right).$$

Logarithmic returns are used widely in finance where they are employed to compare two or more variables when the originating price series consist of highly unequal values [Hudson and Gregoriou 2015].

The *log return* α of a subsequence s of length n is

$$\alpha(s) = \log_{10}\left(\frac{o_n}{o_0}\right).$$

The *unit log return* β of a subsequence s of length n is

$$\beta(s) = \frac{1}{n}\,\alpha(s).$$

The length of a subsequence is important because for many optimisation problems the execution times of the low-level heuristics and objective function evaluations can be non-trivial. The set of all finite length subsequences is denoted S. The unit log return β is used to order the subsequences $s \in S$, so that subsequences with a low β are considered to be superior to subsequences with a high β.

4 VISUALISING ONLINE LEARNING

As described in Section 2, the primary method for understanding hyper-heuristic function is to visualise the learned probabilities of heuristic usage. This shows the learned effectiveness of each heuristic over the course of the optimisation run and can be used to provide insight for domain experts as to which heuristics are the most effective. With the SSHH method, a further level of understanding is possible in that the developed HMM model can be interrogated to determine overall progress and to understand the relationships between heuristics and the acceptance criterion. Two methods for this are described below.

4.1 Progress visualisation using the Kullback Leibler Distance

The effects of online learning can be visualised using the concept of probabilistic length and the KL-distance. Specifically, the effects of online learning on SSHH's HMM can be illustrated by plotting the log number of iterations performed by SSHH against:

(1) the probabilistic length of each state transition vector in the HMM,
(2) the KL-distance between the current HMM and the initial equiprobable HMM,

[2]Objective functions that can produce negative or 0 values must be suitably transformed so as to remove them.

(3) the KL-distance between the HMM before and after online learning, and
(4) the objective function value.

As SSHH employs an identity heuristic emission matrix, increasing the certainty of the next hidden state to be selected is equivalent to increasing the certainty of the next heuristic selection. As certainty increases, so does the probabilistic length of the corresponding state transition n-vector. Plotting the KL-distance between the current HMM and the initial, equiprobable HMM shows how much learning has taken place overall, while plotting the KL-distance between the current HMM before and after online learning shows how much has been learned during a particular learning episode. These three plots are compared with a plot of the log number of iterations against the objective function value of the best solution found so far. This plot shows the overall progress of the optimisation process. As online learning episodes tend to occur less frequently as the optimisation process proceeds, a log scale for iterations is used to improve plot clarity.

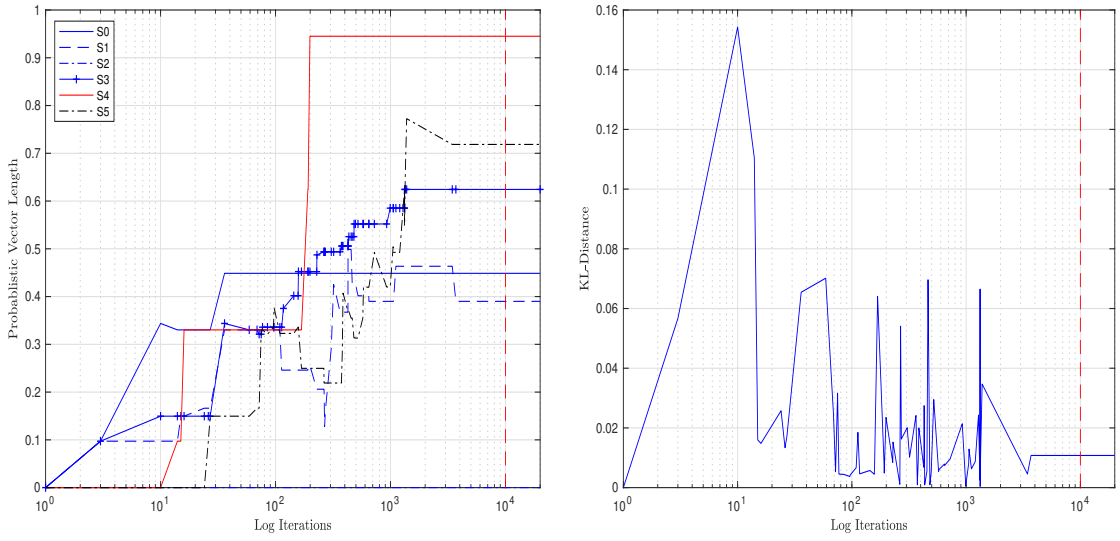
Figure 2 shows a typical run of SSHH on the NYT water distribution network problem [Yates and Keedwell 2021]. In this example, an SSHH hyper-heuristic with six low-level heuristics is executed for 20000 iterations, and finds a minimum solution after 3699 iterations, and 68 online learning episodes. After this point no further online learning takes place and so the traces for probabilistic length, the KL-distances, and the objective function value remain flat. Figure 2a demonstrates that, in this case, although online learning tends to increases the certainty of the next heuristic selection as one would expect, some hidden states such as $S1$ and $S5$ can "forget" and revert to less certain selection probabilities. The general increase in certainty is reflected in Figure 2d that shows that, overall, online learning consistently increases the HMM distance from the initial equiprobable state. As the distance from the equiprobable state increases, so the propensity of the HMM to produce certain subsequences of heuristics over others increases. The changes between HMM states evident in Figure 2b indicate that although the largest changes between states occur early on in the optimisation process, significant changes also occur later on. This supports the hypothesis, that the effectiveness of heuristics and subsequences of heuristics can change during the optimisation process.

4.2 Visualisation of Probability Matrices

Whilst the KL-divergence plots provide an overview of optimisation progress from the model perspective, it is also possible to undertake a deeper investigation of the probability matrices in SSHH. This can provide an in-depth understanding of heuristic utility, combinations of heuristics and the potential for intensification or diversification behaviours for a given problem.

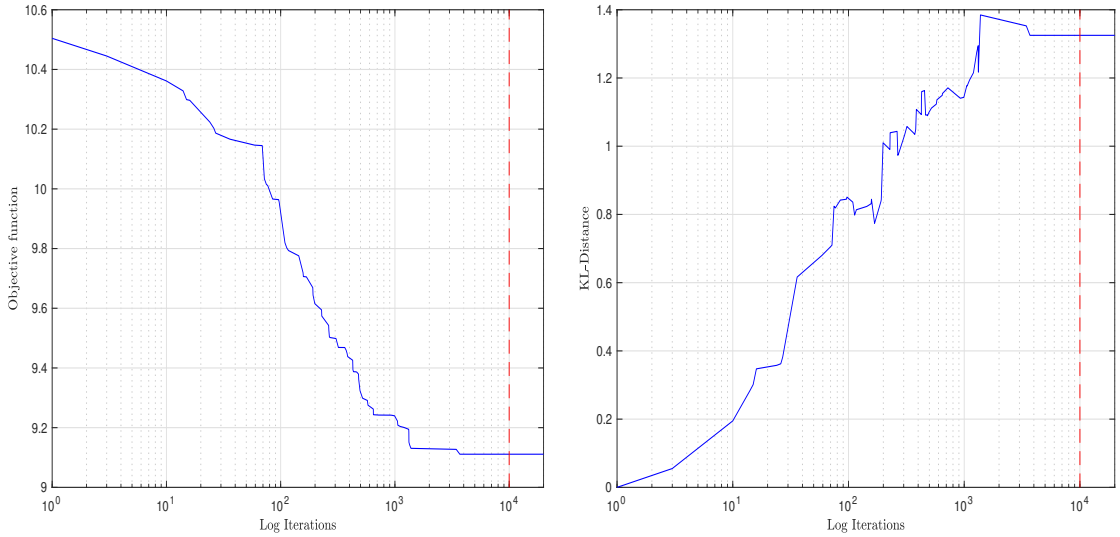
Figure 3 shows the average transition and acceptance check probabilities while solving a travelling salesman problem (TSP) instance within HyFlex [Kheiri and Keedwell 2015]. HyFlex, (Hyper-heuristics Flexible framework [Ochoa et al. 2012]), is a tool that was first proposed in [Woodward et al. 2008] and has been developed to facilitate the development of selection hyper-heuristics, containing an implementation of several optimisation problems along with the implementation of several low-level heuristics. The plots in this figure are averaged (at the end of a run) over 10 runs, with the algorithm run lasting 10 minutes each time, in line with the CHeSC 2011 competition guidelines. HyFlex encapsulates the problem definitions and does not reveal specific problem instances to the user. This design supports the implementation of a single hyper-heuristic that can be applied across multiple problem instances and various problem domains. Therefore, while we know the problem type (TSP), we do not have details on the specific instance solved.

From the acceptance check matrix it can be seen that LLH8 has a high chance to *end* the sequence, and several low-level heuristics have high transition probabilities to move to LLH8 as indicated in the transition matrix in black.



(a) Probabilistic length of each state vector.

(b) KL-distance between HMM states.



(c) Log objective function value.

(d) KL-distance from initial HMM state.

Fig. 2. The online learning traces generated by SSHH when optimising NYT (run 2) resulting in a cost of 44.8854 and resilience 0.4147.

This indicates that this low-level heuristic is contributing strongly to the production of the best solutions, whether performed individually or in sequences. More specifically, a successful sequence of length two that combines LLH5 (which has high probability as seen from the acceptance check matrix) and LLH8 can be seen. A simulation of SSHH operation using these probabilities reveals the following top three sequences: (i) LLH5-LLH8, (ii) LLH8, (iii) LLH0-LLH8.

Even though LLH8 is a successful heuristic, it achieves this good performance with the support of other low-level heuristics (e.g. LLH5 and LLH0) that might seem to have low contribution to the best solutions, but are necessary for overall success. A further investigation shows that LLH5 (and also LLH0) are perturbation operators, and LLH8 is a hill climber (local search algorithm). These combinations are typical in evolutionary computation, where perturbation is applied first to explore the search space and support the *diversification* phase, followed by the application of local search procedures to exploit the potential region of the search space and support the *intensification* phase. These observations show the capability of the SSHH method in identifying good sequences and understanding the relationships between low-level heuristics, and the visualisation of this space demonstrates how the low-level heuristics operate together to deliver near-optimal solutions.

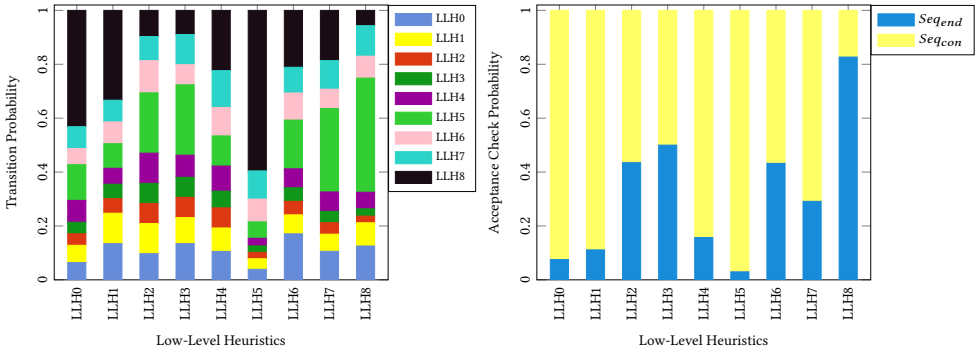


Fig. 3. Transition and sequence construction frequency matrices for TSP [Kheiri and Keedwell 2015]

Visualisation of the emission matrix provides further understanding of the role of heuristic parameter setting in the optimisation [Kheiri and Keedwell 2015]. Each low-level heuristic in HyFlex has an associated parameter that controls its behaviour and takes a value in the range of [0, 1]. In [Kheiri and Keedwell 2015], these parameters are discretised into 11 different parameters: {0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. The specific behaviour of each heuristic with respect to this parameter is not disclosed by HyFlex. This intentional design choice encourages the development of hyper-heuristics that can adaptively control these parameters without explicit knowledge of their internal workings. Our approach relies on observing the outcomes produced by applying these heuristics with different parameter values. After executing a heuristic with a specific parameter, we record the resulting objective value and whether it improves the previous solution. This feedback guides the hyper-heuristic in adjusting the parameters to enhance performance over time. Figure 4 shows the average parameter probabilities from 10 runs while solving a bin packing problem (BP) instance within HyFlex. From this it can be seen that for a set of low-level heuristics (e.g. LLH2, LLH5 and LLH6) large values are preferable, and smaller values are preferable for LLH0 and LLH3. Whereas for others, such as LLH1, the probabilities are distributed equally.

In [Kheiri 2020] SSHH was used to solve a large scale inventory routing problem (IRP), which has a very large search space with many decision variables. In this work, a new emission matrix has been introduced, utilising extended domain information that nonetheless manages low-level heuristics in a domain-independent manner. Figure 5 shows the newly introduced matrix, referred to as solution parameter S, while solving an IRP problem instance. The aim is to learn whether for the selected low-level heuristic to modify the same part of the solution that has been modified during the application of a given sequence (when $S = 0$), or applying the selected heuristic to any part of the solution (when

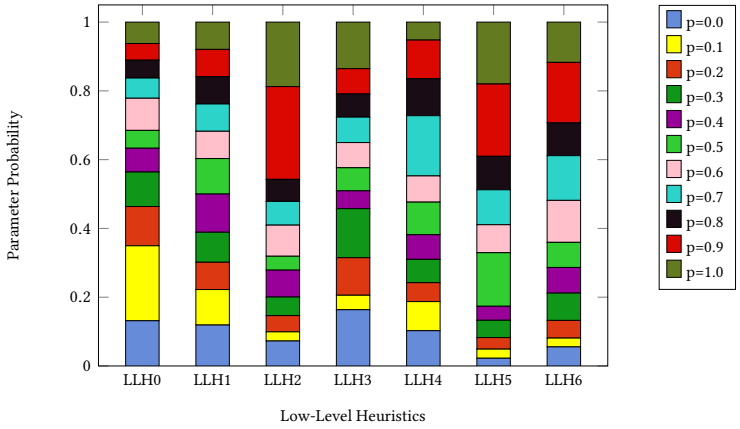


Fig. 4. Probabilities of the parameter matrix for BP [Kheiri and Keedwell 2015]

$S = 1$). While solving the instance, it was observed that applying the sequence of low-level heuristics, starting with LLH0 followed by LLH18, produces high-quality solutions. Figure 5 shows clearly that for this sequence, LLH0 is likely to be applied to a randomly selected part of the solution, while LLH18 is likely to modify the part of the solution that has been altered with LLH0.

4.3 Discussion

The methods described provide visualisation of a complex learning process, distilling the changing set of probabilities into easy to interpret figures. Visualising the learning in this way allows for an understanding of the interplay between low-level heuristic utility during the learning process. The K-L distance also allows the user to understand the progress that is being made by the learning algorithm over time in a similar way to other machine learning algorithms such as the loss curve for neural networks. These visualisations can help to identify anomalies in the training behaviour of the algorithm and to gauge the level of progress made. The more granular visualisation of the low-level heuristics can identify trends in the utility of heuristics, and in particular whether certain heuristics maintain their performance or are 'forgotten' by the hyper-heuristic.The matrix visualisations reveal the interaction between low-level heuristics, their parameters and acceptance strategies. This information can be used to understand the formation of sequences within the algorithm through the transition probabilities and the potential for the objective function to be calculated. It can be used to break down the traditional understanding of which operators promote exploration and intensification, and to develop an understanding of new operators for whom these characteristics might not be known. This form of visualisation can also aid in the fine tuning of heuristic sets and the synergistic operation of pairs or longer sequences of heuristics for specific domains.

5 OFFLINE SUB-SEQUENCE ANALYSIS

The previous sections have focused on explainability within a hidden-markov model based hyper-heuristic. The following sections describe methods for explainability more generally which can operate on any sequences of heuristic invocation, regardless of how they have been generated. In [Yates and Keedwell 2019], the authors present a statistical framework for the analysis of ordered subsets or *subsequences* of low-level heuristics based on logarithmic returns. The

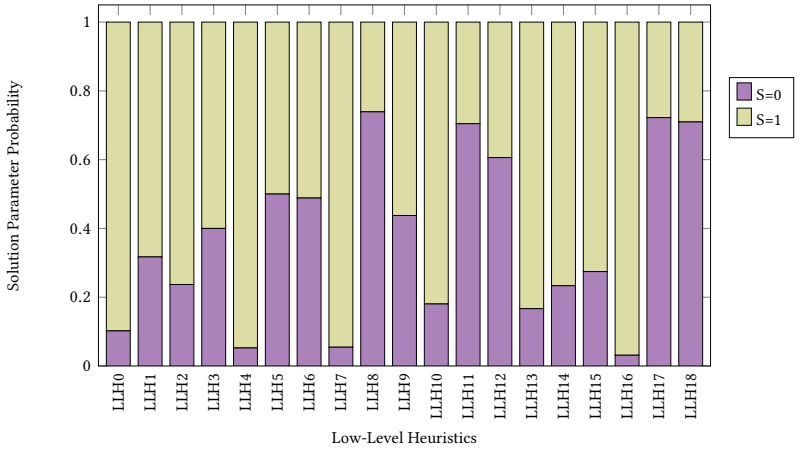


Fig. 5. Probabilities of the solution parameter matrix for IRP [Kheiri 2020]

role of subsequences in the search and optimisation process is investigated via the generation and analysis of a large database of perturbation operations across a number of problems and problem domains. The results of this analysis demonstrate that it is possible to statistically separate "effective" subsequences of perturbations, which tend to decrease the objective function value which is to be minimised, from "disruptive" subsequences of perturbations, which tend to increase the objective value, across most, but not all, problem domains.

Such effective subsequences can be used to improve hyper-heuristic performance either directly, by embedding them in a suitable hyper-heuristic design, or indirectly as the inputs to an appropriate hyper-heuristic learning algorithm.

The statistical approach presented there establishes that subsequences of perturbations are important structural parts of the heuristic search space and that successful selection hyper-heuristics must consider the context in which a low-level heuristic is to be executed before making their selection. The following section explains how such a statistical analysis can be applied to heuristics throughout the lifetime of an optimisation run to determine the volatility of heuristics in solving a particular problem, which in turn will dictate heuristic selection and to what extent offline learning can be used to solve a particular problem type.

5.1 Utility through Time

It is well known that some low-level heuristics are better suited to certain stages or phases of the optimisation process than others. For example, the work of [Remde et al. 2007] on workforce scheduling, that of [Soria-Alcaraz et al. 2014] on course timetabling, and [Yates and Keedwell 2021] on the optimisation of water distribution networks demonstrates that some heuristics which are ineffective at the start of the search process prove to be highly effective at the end, and *vice versa*, while others heuristics are mainly used at the beginning, middle or end of the process.

This section reproduces a method for capturing the differences in heuristic utility that occur during distinct periods of the optimisation process first presented in [Yates and Keedwell 2021]. The water distribution network optimisation problem, used here as an example, is a combinatorial optimisation problem requiring the optimisation of discrete diameter sets usually to minimise cost and pressure constraint violations or to maximise resilience. This real-world problem has many benchmarks available that can vary from tens to thousands of pipes and 10-20 pipe diameters available.

Consider a sequence of low-level heuristics selections, and the associated objective function values generated when optimising a given problem. The *current objective function* value for a heuristic (or subsequence of heuristics) in a run is the objective function value o_t at time t prior to applying the heuristic (subsequence). The sets HIGH and LOW consist of all those heuristic instances which have a current objective function value

$$o_t > P^r_{90} \text{ and } o_t < P^r_{10} \tag{1}$$

respectively, where P^r_{90} is the 90th and P^r_{10} is the 10th percentile. The HIGH and LOW sets contain heuristic occurrences that occur at the "beginning" of the optimisation process, when objective function values are relatively high, and at the "end" of the optimisation process, when objective function values are relatively low. However, it should be noted that in the absence of elitism in the optimisation process, the relationship between time and objective function value is not necessarily linear. Calculating the percentile values over the heuristic sequences generated when optimising a number of problems can lead to heuristic occurrences from a few problems dominating the sets; those that produce very high or very low objective function values. Here the percentiles P^r are calculated locally over the objective function values of *each* optimisation run $r = 1, \ldots, N$ on each problem. This ensures that heuristic instances from all the problems are included in the sets.

The heuristic instances can be further subdivided according to their current objective function values into 10 sets

$$P_{10} = [P^r_{90}, P^r_{100}],\ P_9 = [P^r_{80}, P^r_{90}], \ldots, P_1 = [P^r_0, P^r_{10}].$$

The utility of a heuristic (subsequence) can be estimated in a number of ways (see for example [Fialho et al. 2008; Özcan et al. 2010; Soria-Alcaraz et al. 2014; Yates and Keedwell 2019, 2021]). In this study the utility of a heuristic is measured by the mean log return $\bar{\alpha}$ of a set of N occurrences of a given heuristic s defined by

$$\bar{\alpha}(\{s^1, \ldots, s^N\}) = \frac{1}{N} \sum_{i=1}^{N} \alpha(s^i).$$

Negative values of $\bar{\alpha}$ indicate an effective heuristic, that tends to reduce the objective function value, while positive values of $\bar{\alpha}$ indicate a disruptive heuristic, that tends to increase the objective function value. In this classification, heuristics that have a $\bar{\alpha}$ that is zero, or close to zero, can be regarded as neutral. By way of contrast, in [Yates and Keedwell 2021] heuristic utility is measured by the *contribution* of the heuristic to the optimisation process, which is defined to be the proportion of the decrease (or increase) in the objective function value due to that heuristic.

By calculating a given measure of utility for each heuristic, over a number of optimisation runs, and each percentile P^r it is possible to estimate the relative utility of each heuristic during the optimisation process.

The heuristics in a percentile P^r can be ranked by their utility values. The change in utility order between percentiles can be quantified by using the Spearman's Footrule distance [Diaconis and Graham 1977]. The distance is calculated by taking the sum of the absolute values of the difference between two ranks. In symbols, if σ and π denote two permutations of n elements such that $\sigma(i)$ and $\pi(i)$ denote the rank of an element $i = 1, \ldots, n$ in the permutation, then Spearman's Footrule is defined by

$$d(\sigma, \pi) = \sum_{i=1}^{n} |\sigma(i) - \pi(i)| \tag{2}$$

and has a maximum integer value of $m = \lfloor \frac{1}{2} n^2 \rfloor$ where $\lfloor \cdot \rfloor$ is the floor function. The greater the Footrule distance, the greater the difference between the two orders. Large Footrule distances between percentiles indicate that the relative utility of the low-level heuristics changes significantly during the optimisation process. This has important implications

for online and offline learning. For example, when the change in heuristic utility is large, it is to be expected that online learning can improve optimisation performance by allowing the hyper-heuristic to select the most effective heuristics at any given point in time. It would also render offline learning less effective, as the learned sequences would only be useful at particular points during the optimisation process.

Table 2. The WDN heuristics in the HIGH and LOW sets ordered by ascending mean log return $\bar{\alpha}$, the Spearman's Footrule distance d, and the normalised Footrule distance $d' = \frac{d}{m}$.

Problem Name	Abbr.	Set	$\bar{\alpha}$-order (asc. left to right)	d	d'
Two-Reservoir	TRN	HIGH	$C_5 < R_3 < S_4 < M_0 < S_1 < M_2$	10	0.56
		LOW	$C_5 < M_2 < M_0 < R_3 < S_1 < S_4$		
New York	NYT	HIGH	$C_5 < S_4 < R_3 < S_1 < M_0 < M_2$	12	0.67
		LOW	$C_5 < M_2 < M_0 < R_3 < S_1 < S_4$		
Fossolo	FOS	HIGH	$S_4 < C_5 < R_3 < S_1 < M_0 < M_2$	16	0.89
		LOW	$M_2 < M_0 < C_5 < S_1 < R_3 < S_4$		
Modena	MOD	HIGH	$C_5 < R_3 < M_0 < M_2 < S_1 < S_4$	6	0.33
		LOW	$M_2 < C_5 < M_0 < R_3 < S_1 < S_4$		
Exeter	EXN	HIGH	$C_5 < R_3 < M_0 < M_2 < S_1 < S_4$	6	0.33
		LOW	$C_5 < M_2 < M_0 < S_1 < R_3 < S_4$		
All	ALL	HIGH	$C_5 < S_4 < R_3 < S_1 < M_0 < M_2$	14	0.78
		LOW	$M_2 < C_5 < M_0 < S_1 < R_3 < S_4$		

For example consider the 5 water distribution network problems TRN, NYT, FOS, MOD, and EXN presented in [Yates and Keedwell 2021]. The results in table 2 show the low-level heuristics in the HIGH and LOW subsets ordered by ascending mean log return $\bar{\alpha}$. The ALL row shows the HIGH and LOW subsets averaged over all 12 problems. The Footrule distances indicate some large differences in the orderings of the LOW and HIGH heuristic sets. For example, notice how the M_2 heuristic changes from being one of the least effective heuristics in the HIGH sets, to one of the most effective heuristics in the LOW sets. These large differences in rank indicate that different individual heuristics are effective at different points in the optimisation process. Figure 6 shows the effectiveness of the low-level heuristics for each percentile over all 12 problems; where M0 – change one pipe diameter randomly, S1 – swap two pipe diameters at random, M2 – increase or decrease a randomly selected pipe diameter by one size, R3 – "ruin" several (1-5) pipes and rebuild randomly, S4 – shuffle several (1-5) pipes (i.e. makes several swaps) and C5 – two-point crossover of two vectors of network pipe diameters. The plot illustrates the changes in heuristic effectiveness as solution optimality increases. Notice that the two-point crossover heuristic C_5 is the best performing low-level heuristic in all but the last two percentiles P^r_{20} and P^r_{10} and S_4 performs well initially but is the worst performer from P^r_{80} onwards. These performance characteristics align with expectations of operator behaviour, for example S_4 is a disruptive shuffle operation which is likely to make significant gains when objective function values are poor, but is disruptive to better performing solutions as the optimisation progresses. Figure 7 shows the heuristic effectiveness for each heuristic class. The bars show the maximum and minimum $\bar{\alpha}$ values which demonstrate the ability of each heuristic to generate improving or worsening solutions. The shuffle operators (Figure 7a) show the greatest variation as might be expected, whereas the mutation operators (Figure 7d) show divergent behaviour. Operator M_2 demonstrates consistency at all objective function levels, generating fewer strongly improving moves but also fewer strongly worsening moves. This behaviour sees it move from being the worst-ranked heuristic in P^r_{100} to the best ranked in P^r_{10}.

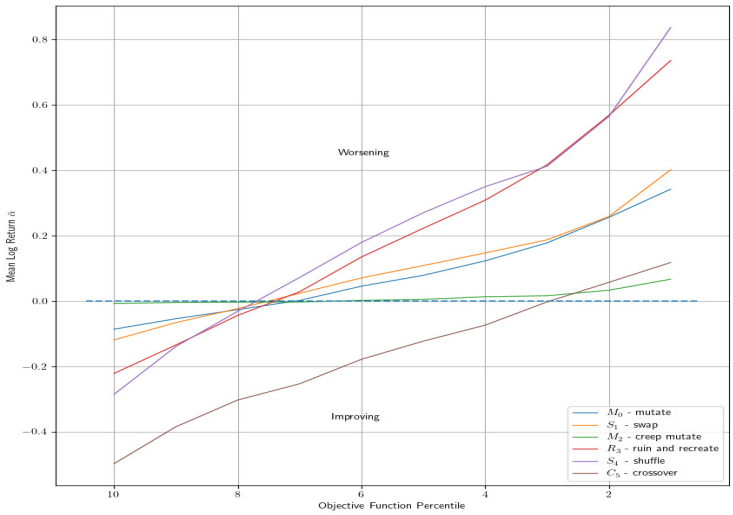


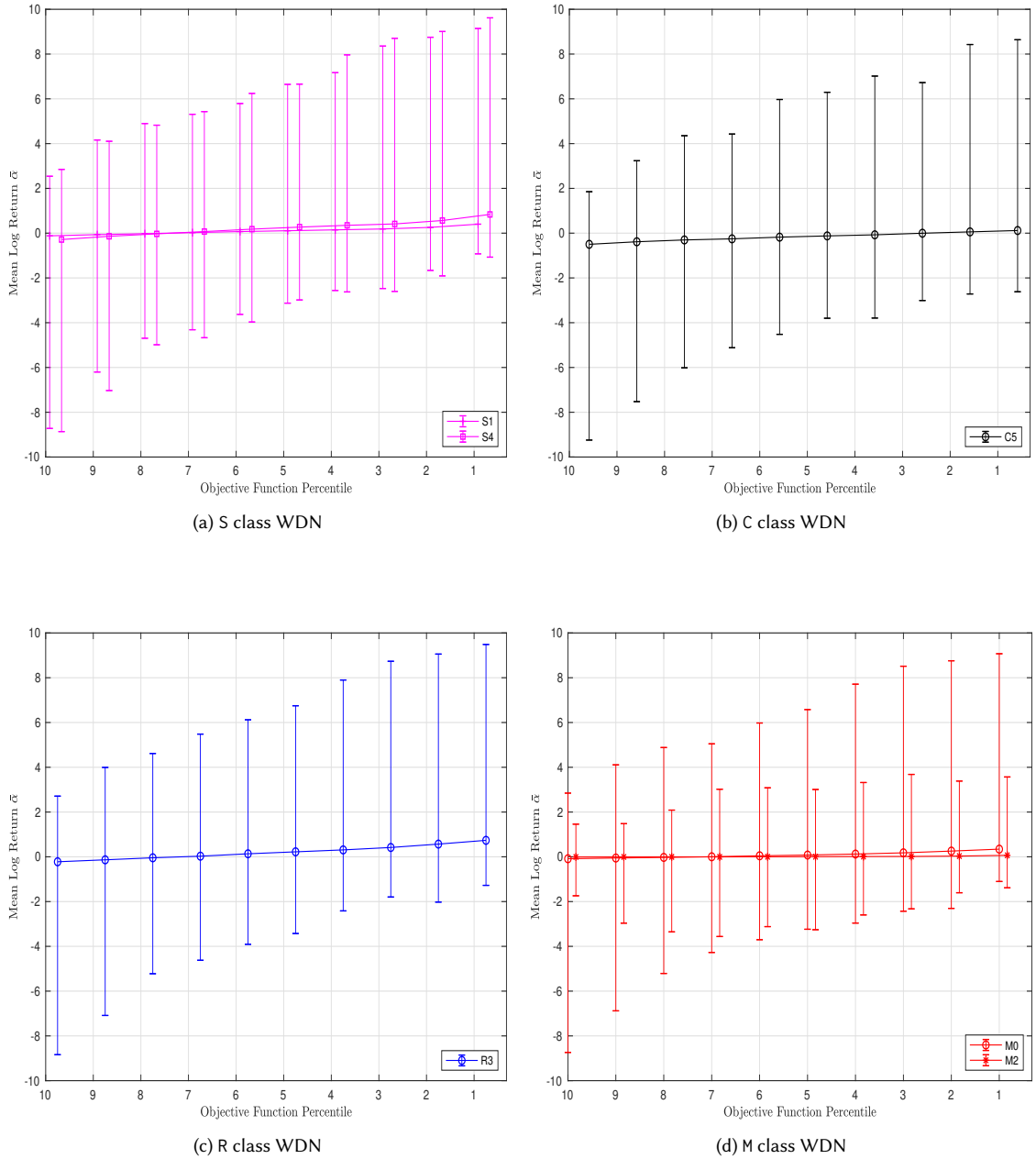
Fig. 6. The mean log return $\bar{\alpha}$ of the WDN low-level heuristics averaged over 10 local percentiles. Optimality increases from left to right, while negative $\bar{\alpha}$ values correspond to reductions in the objective function value.

5.2 Discussion

Visualising individual heuristics and their efficacy at different stages during the optimisation process is an important method of understanding hyper-heuristic behaviour. It is especially important for offline methods where due consideration should be given to when to retrain the offline method during the optimisation. It is also important for online methods which may not retain sufficient plasticity later in the search to enable a switch from an already learned set of heuristic probabilities to another. Therefore the algorithm design should consider the likely change in heuristic utility on a given problem before considering which approach will be taken. However, it is clear from Table 1 through an understanding of the differences present in heuristic sequences (using Spearman's footrule as a metric), that there is potentially significant variation among problem instances even within the same domain.

6 CLUSTERING OF SEQUENCES

Thus far the focus has been on explaining the execution of the hyper-heuristic and the information learned during optimisation. Cluster analysis methods can be used to form an understanding of the problems and domains from an algorithmic rather than a subjective perspective. Typically problems and their difficulty are described in terms of the number and type of decision variables, objective function complexity and the difficulty (e.g. multimodality) of the resulting search space. However, for hyper-heuristics, these complexities are mediated by the number and type of available low level heuristics and so methods of explainability can exploit the interface between algorithm and problem. In [de Hoon et al. 2008; Duda et al. 2001] the *k-medoids* and the *pairwise average-linkage* clustering algorithms are used to explore this potential for explaining the algorithm-problem nexus.



(a) S class WDN

(b) C class WDN

(c) R class WDN

(d) M class WDN

Fig. 7. The mean log return $\bar{\alpha}$ of the low-level heuristics in the WDN domain, averaged over the 10 local percentiles. The optimisation process proceeds in time from left to right, while negative $\bar{\alpha}$ values correspond to reductions in the objective function value. The bars show the maximum and minimum $\bar{\alpha}$ values.

The k-medoids algorithm is related to the k-means algorithm. Both the k-means and k-medoids algorithms partition the data set into groups and both attempt to minimise the distance between points labeled to be in a cluster and a point designated as the centre of that cluster. For the k-means algorithm, this point need not be an element of the set of data points being clustered. In contrast, the k-medoids algorithm must use data points as centres (or medoids). A pairwise average-linkage hierarchical clustering algorithm is used to construct binary trees. In average-linkage clustering, the distance between two nodes is defined as the average over all pairwise distances between the elements of the two nodes. The results of cluster analysis and the hierarchical cluster diagrams can facilitate the identification of similarities and differences that occur when a selection hyper-heuristic is executed on a number of given problems (and domains).

6.1 The Smith-Waterman Algorithm

Methods are required to compare sequences of heuristics to obtain an understanding of the problem space from an algorithmic perspective [Yates and Keedwell 2017]. However, the comparison of sequences is not straightforward. For example, using the Hamming distance, two identical binary strings will appear dissimilar, that is, the Hamming distance will be large if one string is shifted by one character in either direction. The Smith-Waterman algorithm [Smith and Waterman 1981; Zhao et al. 2013] overcomes this problem by attempting to identify similar regions of any given pair of strings. In bioinformatics, the Smith-Waterman algorithm is used to analyse the arrangement of DNA/RNA or protein sequences. The algorithm performs a *local sequence alignment* by use of dynamic programming; instead of considering the whole sequence, the algorithm compares subsequences of all possible lengths and optimises a similarity measure. A large similarity score produced by the algorithm implies that the strings are very similar. A similarity score of 0 implies that the two strings have no symbols in common.

The similarity measure is defined by a *similarity matrix* and a set of *gap penalties*. The similarity matrix defines the positive score for matching two symbols or the cost of mismatching two symbols. The gap penalties specify the score or cost of opening up a gap in a string and extending that gap in order to improve the fit with another string. Although the similarity matrix and gap penalties can be adjusted to alter the behaviour of the algorithm, in general it is not known which values are best suited for optimisation problems. As an example, consider the similarity matrix

	L	C	R	M
L	3	−2	−2	−2
C	−2	3	−2	−2
R	−2	−2	3	−2
M	−2	−2	−2	3

defined over the alphabet $\{\mathtt{L}, \mathtt{C}, \mathtt{R}, \mathtt{M}\}$ employed in [Yates and Keedwell 2017], where the gap open and gap extend penalties are −3 and −1 respectively. Given the target sequence (where M=mutation, C=crossover, R=ruin and recreate, L=local search):

MCLRLRMRRRRLRMCMRRRRLLMRCMMRLMRRRRCRLLL

and the query sequence

LRRCMCLMLLLLL

the best match produced by the Smith-Waterman algorithm is shown in table 3. In this case there are 9 matches, 2 mismatches, 2 gaps, and 5 gap extends giving a score of $(9 \times 3) + (2 \times -2) + (2 \times -3) + (5 \times -1) = 12$.

Table 3. The output produced by the Smith-Waterman algorithm for the target and query sequences where the characters '|' denotes a match, '*' denotes a mismatch, and '–' denotes a gap. The Smith-Waterman score is 12.

Target	22	LMRCMMRLMRRRRCRLLL	39
		|*|| |*|| |||	
Query	1	LRRC-MCLM————LLL	11

The Smith-Waterman algorithm can be used to construct a simple notion of distance d between sequences. In symbols

$$d(s_1, s_2) = \frac{sw(s_1, s_2)}{\sqrt{sw(s_1, s_1)\, sw(s_2, s_2)}}.$$

A low d value indicate that two sequences are similar or close. This function should only be loosely interpreted as a distance function as it is not a metric in the formal sense, but for the purposes of explaining algorithm decisions it provides a method to understand how similar two heuristic sequences are.

6.2 Cluster Analysis

A k-medoid clustering algorithm employing the Smith-Waterman distance can be used to separate an offline learning database of heuristic sequences into a number of clusters. For clustering purposes, only the sequence selections up to and including the minimum objective function value are used, as these are the selections that would normally be used as learning algorithm inputs. The accuracy of the resulting clusters can be evaluated using the four commonly used measures: *purity*, *normalised mutual information (NMI)*, *Rand index*, and the F_5 [Manning et al. 2008]. For each measure, the worst clusterings have values close to 0 while a perfect clustering has a value of 1.

The cluster centre points (medoids) are the "typical" sequences for that cluster. A pairwise average-linkage hierarchical clustering algorithm can be used to construct a binary tree of the cluster centres. When the clusters represent different problems (or domains), and are well defined, the resulting hierarchical cluster diagrams can be used to identify similarities and differences between problems and problem domains.

The two experiments presented in Sections 6.3 and Section 6.4 demonstrate that the sequences of heuristics produced by a selection hyper-heuristic contain subsequences that (in some cases) are common to a particular problem instance and that differ significantly between (some) problem domains. The existence of discernible subsequences of heuristics lends weight to the argument that the ordering of a subsequence of heuristics is crucial to search efficacy and this ordering varies with problem and problem domain. The ability to identify (similar) problems and domains from a sample of heuristic selections can be used to guide the choice of learning algorithm and learning algorithm parameters for unseen problem (domains) or those with novel heuristic sets without requiring problem specific information. These results also demonstrate the suitability of the Smith-Waterman algorithm as a measure of sequence similarity for offline learning applications.

6.3 Separating Problem Domains

This section contains an example, first presented in [Yates and Keedwell 2017], of using clustering algorithms to separate a number of distinct problem domains.

HyFlex [Ochoa et al. 2012]) is again used here (see Section 4.2). The 4 domains employed here are:

(1) 1D bin packing (BP),
(2) permutation flow shop (PFS),

(3) boolean satisfiability (SAT), and
(4) personnel scheduling (PS).

Each problem domain contains 10 distinct problems of varying complexity. HyFlex hides all problem specific information such as the solution representations, the solution constructions, and the low-level heuristic implementations. Each HyFlex domain has four general classes of low-level heuristic:

(1) mutation (M) which perturbs a solution randomly,
(2) crossover (C) which constructs a new solution from two or more existing solutions,
(3) ruin and recreate (R) which destroys a given solution partially and then rebuilds the deleted parts, and
(4) local search (L) that incorporates an iterative improvement process and returns a non-worsening solution.

The actual number and implementation of the low-level heuristics differs between problem domains. A simple hyper-heuristic is executed for 150 selections, 40 times on each of the 10 HyFlex problems in each of the 4 domain. The resulting $40 \times 10 \times 4 = 1600$ sequences of 150 heuristic class selections and objective function values are used to construct the offline database.

A k-medoid clustering algorithm employing the Smith-Waterman distance is used to separate the entire offline learning database of 1600 sequences into 4 clusters corresponding to the 4 HyFlex domains. The accuracy of the resulting clusters are shown in table 4 and they demonstrate that the Smith-Waterman distance is able to form "good" clusters. The metrics used within this table are commonly used metrics of clustering performance [Manning et al. 2008]. Purity uses the class label for each class and divides the number of correct datapoints in each cluster by the total, a method similar to an accuracy calculation in classification. NMI (normalised mutual information) uses an entropy calculation between clusters and delivers a maximal value of 1.0. The Rand Index is widely used in statistics and effectively divides the sum of the true positives and true negatives by the sum of all datapoints in the set. The F5 metrics penalises false negatives more strongly than false positives and thus gives more weight to recall.

Table 4. Clustering accuracy for the 4 HyFlex domains.

Purity	NMI	Rand	F_5
0.8269	0.5954	0.7951	0.8001

Figure 8 shows the result of applying an average-linkage hierarchical clustering algorithm to the 1600 sequences of the offline learning database. As the tree obtained by clustering all 1600 sequences is too large to include here, the sequences were broken down into 40 sets of sequences where each set contains the 40 runs for that problem. For example, sequence set 1 consists of the 40 runs of problem 1 (domain 1), while sequence set 2 consists of the 40 runs of problem 2 (domain 1), and so on. It is clear from Figure 8 that the clustering algorithm when using the Smith-Waterman distance is able to separate the 4 underlying domains. In fact, it is quite remarkable that the method is able to distinguish each of the four domains so accurately with only one outlier from heuristic sequences only. Furthermore this plot can be used to identify outlier instances (e.g. instance 20) and to understand that PFS, SAT and PS are more closely related to each other, than the 1D bin packing problem, from the algorithm's perspective.

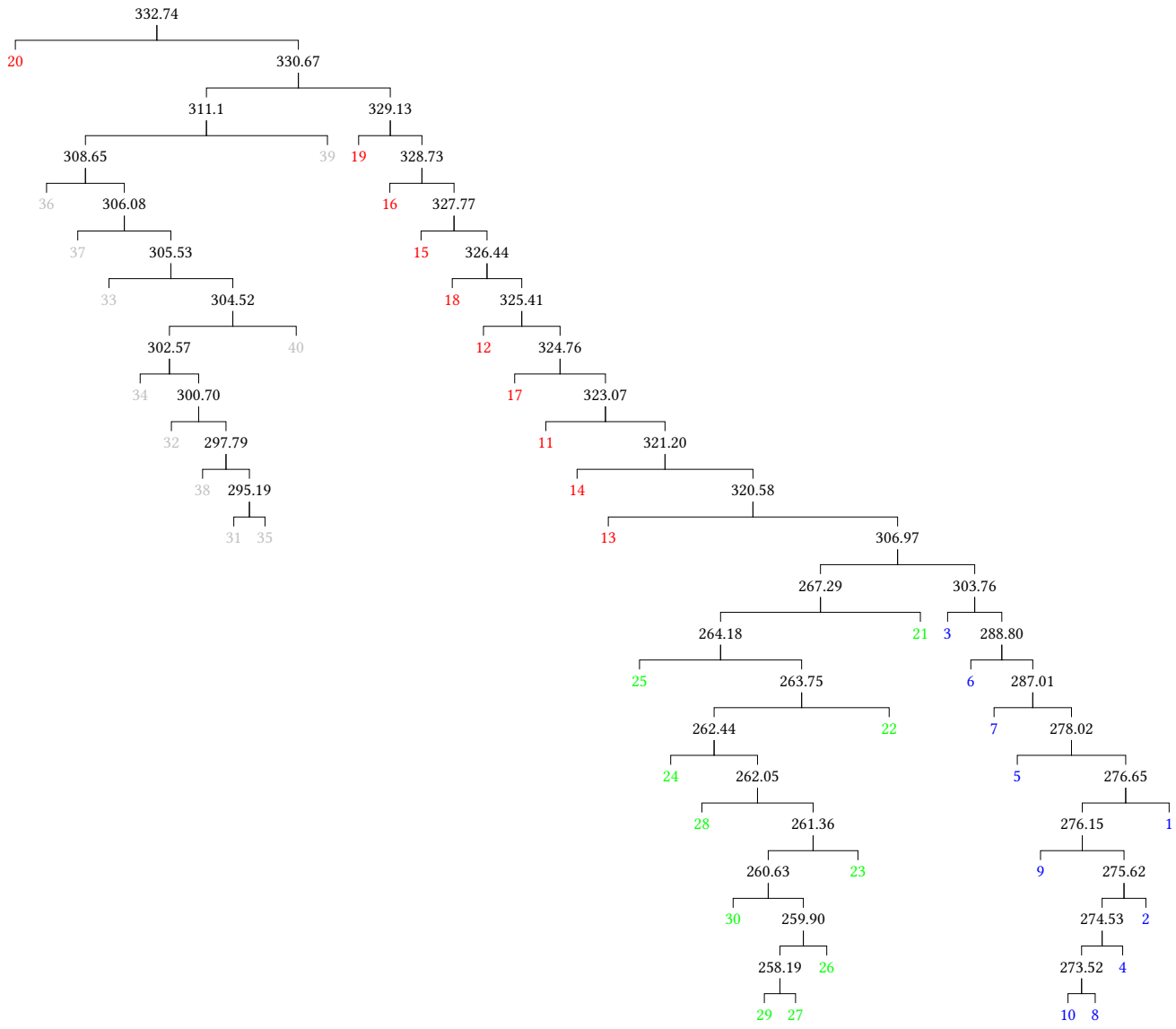
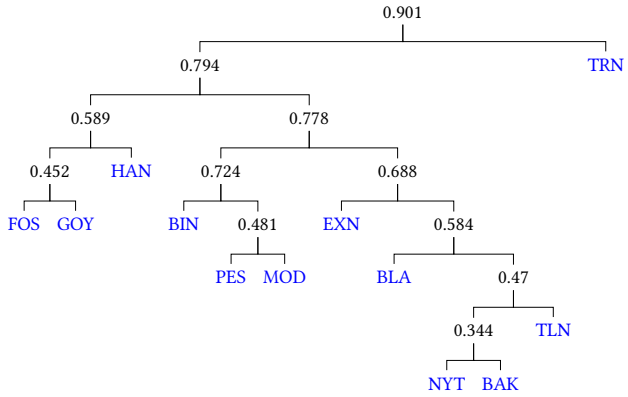


Fig. 8. The result of average-linkage hierarchical clustering on the 40 sets of 40 sequences. Each leaf node identifies a sequence set consisting of the 40 runs for that HyFlex problem. The blue labeled sequences are drawn from domain 1, the red labeled sequences are drawn from domain 2, the green labeled sequences are drawn from domain 3, and the grey labeled sequences are drawn from domain 4.

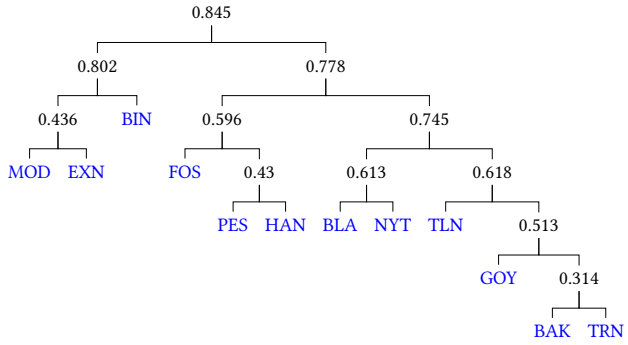
6.4 Separating Problems

This section presents the results of applying clustering algorithms to a number of Water Distribution Network problems [Yates and Keedwell 2021]. Here the objective is to investigate how the choice of low-level heuristics affects the optimisation process.

In this example, the results of two average-linkage hierarchical clusterings on the $12 \times 40 = 480$ sequences produced by SSHH on the 12 water distribution network problems [Yates and Keedwell 2021] are shown in Figure 9. For the BASE clustering, SSHH is parameterised with the three low-level heuristics $\mathtt{M}_\emptyset$, $\mathtt{S}_1$, $\mathtt{C}_5$, while for the PIPE clustering, the SSHH paramerisation includes an additional "novel" heurisric $\mathtt{P}_6$. The associated accuracy measures for the resulting clusters are shown in Table 5.



(a) Clustering the BASE parameterisation consisting of 3 low-level heuristics.



(b) Clustering the PIPE parameterisation consisting of the 3 BASE low-level heuristics and the extra $\mathtt{P}_6$ heuristic.

Fig. 9. The result of average-linkage hierarchical clustering on the 480 sequences produced by SSHH on the 12 water distribution network problems.

Table 5. A comparison of clustering accuracy. The clustering is performed on the entire set of 480 sequences.

Heuristic Set	Purity	NMI	Rand	F_5
BASE	0.3312	0.3539	0.8335	0.3409
PIPE	0.3708	0.3944	0.8215	0.4076

Figure 9 illustrates that the TRN problem differs significantly from the remaining 11 problems (see figure 9a), and that the differences between problems can change significantly when an extra low-level heuristic is added (see Figure 9b). Such knowledge can be used to improve the optimisation performance of a hyper-heuristic, by, for example, guiding

the choice of optimisation parameters as in [Tsoumakas et al. 2004], the choice of low-level heuristics, or the selection of appropriate training sequences for a machine learning algorithm.

6.5 Discussion

By using clustering and visualisation tools such as these, it is possible to view the problem instance set from the perspective of the algorithm rather than a pre-ordained delineation by problem type as it is understood by domain experts. This is important for a number of reasons. Where the underlying heuristics have common properties (as in Hyflex) it can be used to identify similarities and differences between problem domains that may not be apparent prior to optimisation and are only revealed once optimisation has been conducted. It also allows for a greater understanding of the domains that are most closely related and therefore whether similar algorithm types and heuristic sets might be most appropriate to solve them. In particular, clustering allows us to group together/distinguish problem instances based on the sequences of low level heuristics used to optimise them. This facilitates our understanding of the problems. For example, similar sequences of heuristics could indicate that the problems have similar structural characteristics/properties. It can also help to identify which instances are most typical of their domain and which are outliers, providing greater understanding of the difficulty of each instance and concomitant algorithm performance. Finally, when presented with a new problem, the ability to identify which previously encountered problems are most similar to this new problem provides a potentially better initial starting point for research as opposed to a completely blank slate.

7 CONCLUSION

In this paper we have proposed methods for the understanding of hyper-heuristics as a step towards explainable optimisation through the analysis of sequences of low-level heuristics and the probabilities of selection, coupled with statistical and clustering methods. It should be noted that while the K-L distances and investigation of the transition matrices in section 4 are specific to hidden markov model-based approaches such as 3.1, the investigation of heuristic sequences, utility through time and clustering can be performed on any selection hyper-heuristic providing it selects and invokes low-level heuristics. These analyses enable an understanding not only of the best performing individual heuristics, but also of algorithm progress, the best performing intensification and diversification strategies and an understanding of how these change for different problems and stages of the optimisation lifetime. They also provide an avenue for explainability to feedback into algorithm design. Each of the approaches has the potential to inform changes to the algorithm formulation, parameterisation, algorithm behaviour or research pathway through an in-depth understanding of what works within a particular domain. This is particularly important in an area such as hyper-heuristics where the setup phase of the algorithm is quite often conducted manually and so guidance at this stage as to high-performing heuristic sets or hyper-heuristics for similar domains can be particularly useful.

Developing an understanding of heuristic (operator) utility is the cornerstone of research in hyper-heuristics, landscape analysis and adaptive metaheuristics, and the extension of this to include sequences is an interesting area for further research. Understanding utility through time provides a further dimension for analysis as the effectiveness of perturbations is known to vary during the optimisation, but delivering the most effective adaptation to this is still an open research question. This work has shown that fully explainable optimisation requires analysis throughout the optimisation run, not just through individual snapshots. Further investigations in this area might include a better understanding of the effectiveness of selection and acceptance criteria through time in addition to the perturbation operators. In particular, the parameterisation of selection operators and thresholding of acceptance criteria are also likely to need to adapt to deliver high quality optimisation results and might prove a fruitful further avenue of research.

The clustering of sequences of operation provide an insight into the algorithm-problem nexus and has the potential to break down traditional problem domains. This automated approach has the potential to determine the similarities and differences between problem instances and domains through the lens of the algorithm and heuristics being used, and provides a further level of explanation for hyper-heuristic performance within a domain.

Finally, many of the above tools and those described for metaheuristics are focused on explaining the search behaviour in a way that the algorithm developer can understand and feed into further algorithm design, or to extract best performance from the hyper-heuristics being used. Explaining optimisation decisions for the end-user remains a more difficult problem to solve as typically solutions are not produced by these methods through a systematic approach that can be easily communicated. The metaheuristic or hyper-heuristic method of iterative cycles of perturbation and acceptance decisions would appear to be relatively dissimilar to the way in which a human would approach the same problem, making it difficult to find common ground between the approaches. Further work in this area would certainly be welcomed to explain to an end-user how the final solutions produced by a hyper-heuristic have been produced throughout an optimisation run.

REFERENCES

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* 64 (2013), 1695–1724.

Edmund K Burke, Matthew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. 2019. A classification of hyper-heuristic approaches: revisited. *Handbook of metaheuristics* (2019), 453–477.

Xinan Chen, Ruibin Bai, Rong Qu, Jing Dong, and Yaochu Jin. 2024. Deep Reinforcement Learning Assisted Genetic Programming Ensemble Hyper-Heuristics for Dynamic Scheduling of Container Port Trucks. *IEEE Transactions on Evolutionary Computation* (2024).

Peter Cowling, Graham Kendall, and Eric Soubeiga. 2001. A hyperheuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3*. Springer, 176–190.

M. de Hoon, S. Imoto, and S. Miyano. 2008. *The C clustering library for cDNA microarray data.* Technical Report. Human Genome Center, Institute of Medical Science, University of Tokyo.

P. Diaconis and R. L. Graham. 1977. Spearman's Footrule as a Measure of Disarray. *Journal of the Royal Statistical Society. Series B* 39, 2 (1977), 262–268.

John H Drake, Ahmed Kheiri, Ender Özcan, and Edmund K Burke. 2020. Recent advances in selection hyper-heuristics. *European Journal of Operational Research* 285, 2 (2020), 405–428.

R. O. Duda, P. E. Hart, and D. G. Stork. 2001. *Pattern Classification* (second ed.). Wiley-Interscience.

A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. 2008. Extreme Value Based Adaptive Operator Selection. In *Parallel Problem Solving from Nature – PPSN X*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 175–184.

Geoffrey E Hinton, Steven J Nowlan, et al. 1987. How learning can guide evolution. *Complex systems* 1, 3 (1987), 495–502.

Ping-Che Hsiao, Tsung-Che Chiang, and Li-Chen Fu. 2012. A vns-based hyper-heuristic with adaptive computational budget of local search. In *2012 IEEE congress on evolutionary computation*. IEEE, 1–8.

R. S. Hudson and A. Gregoriou. 2015. Calculating and comparing security returns is harder than you think: A comparison between logarithmic and simple returns. *International Review of Financial Analysis* 38 (2015), 151 – 162. https://doi.org/10.1016/j.irfa.2014.10.008

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*. Springer, 507–523.

B.-H Juang and L. R. Rabiner. 1985. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal* 64, 2 (1985), 391–408. https://doi.org/10.1002/j.1538-7305.1985.tb00439.x

Graham Kendall, Peter Cowling, Eric Soubeiga, et al. 2002. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*. Citeseer, 667–671.

Ahmed Kheiri. 2014. *Multi-stage hyper-heuristics for optimisation problems.* Ph. D. Dissertation. University of Nottingham.

A. Kheiri. 2020. Heuristic Sequence Selection for Inventory Routing Problem. *Transportation Science* 54, 2 (2020), 302–312. https://doi.org/10.1287/trsc.2019.0934

Ahmed Kheiri, Angeliki Gretsista, Ed Keedwell, Guglielmo Lulli, Michael G Epitropakis, and Edmund K Burke. 2021. A hyper-heuristic approach based upon a hidden Markov model for the multi-stage nurse rostering problem. *Computers & operations research* 130 (2021), 105221.

A. Kheiri and E. C. Keedwell. 2015. A Sequence-based Selection Hyper-heuristic Utilising a Hidden Markov Model. In *Proceedings of the Genetic and Evolutionary Computation* (Madrid, Spain). ACM, 417–424. https://doi.org/10.1145/2739480.2754766

A. Kheiri and E. C. Keedwell. 2017. A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper-heuristics with a Case Study in High School Timetabling Problems. *Evolutionary Computation* 25, 3 (2017), 473–501. https://doi.org/10.1162/EVCO_a_00186

Ahmed Kheiri and Ender Özcan. 2016. An iterated multi-stage selection hyper-heuristic. *European Journal of Operational Research* 250, 1 (2016), 77–90.

C. Lu, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin. 2013. A Normalized Statistical Metric Space for Hidden Markov Models. *IEEE Transactions on Cybernetics* 43, 3 (2013), 806–819. https://doi.org/10.1109/TSMCB.2012.2216872

Mashael Maashi, Graham Kendall, and Ender Özcan. 2015. Choice function based hyper-heuristics for multi-objective optimization. *Applied Soft Computing* 28 (2015), 312–326.

Mashael Maashi, Ender Özcan, and Graham Kendall. 2014. A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications* 41, 9 (2014), 4475–4493.

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

Mustafa Misir. 2012. *Intelligent hyper-heuristics: A tool for solving generic optimisation problems.* Ph. D. Dissertation.

Mustafa Misir, Pieter Smet, Katja Verbeeck, and Greet Vanden Berghe. 2011. Security personnel routing and rostering: a hyper-heuristic approach. In *Proceedings of the 3rd International Conference on Applied Operational Research*, Vol. 3. Tadbir; Canada, 193–205.

Mustafa Mısır, Katja Verbeeck, Patrick De Causmaecker, and Greet Vanden Berghe. 2012. An intelligent hyper-heuristic framework for chesc 2011. In *Learning and Intelligent Optimization: 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers*. Springer, 461–466.

Julia Moosbauer, Julia Herbinger, Giuseppe Casalicchio, Marius Lindauer, and Bernd Bischl. 2021. Explaining hyperparameter optimization via partial dependence plots. *Advances in Neural Information Processing Systems* 34 (2021), 2280–2291.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223* (2019).

G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic, and E. K. Burke. 2012. HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search. In *Evolutionary Computation in Combinatorial Optimization*, Jin-Kao Hao and Martin Middendorf (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 136–147.

Gabriela Ochoa and Katherine Malan. 2019. Recent advances in fitness landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1077–1094.

Ender Özcan and Ahmed Kheiri. 2012. A hyper-heuristic based on random gradient, greedy and dominance. In *Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences*. Springer, 557–563.

E. Özcan, M. Mısır, G. Ochoa, and E. K. Burke. 2010. A Reinforcement Learning – great-Deluge Hyper-Heuristic for Examination Timetabling. *International Journal of Applied Metaheuristic Computing* 1, 1 (2010), 39–59. https://doi.org/10.4018/jamc.2010102603

Erik Pitzer and Michael Affenzeller. 2012. A comprehensive survey on fitness landscape analysis. *Recent advances in intelligent engineering systems* (2012), 161–191.

Rong Qu, Edmund K Burke, and Barry McCollum. 2009. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research* 198, 2 (2009), 392–404.

Rong Qu, Nam Pham, Ruibin Bai, and Graham Kendall. 2015. Hybridising heuristics within an estimation distribution algorithm for examination timetabling. *Applied Intelligence* 42 (2015), 679–693.

L. R. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceddings of the IEEE* 77, 2 (1989), 257–286. https://doi.org/10.1109/5.18626

S. Remde, P. Cowling, K. Dahal, and N. Colledge. 2007. Exact/Heuristic Hybrids Using rVNS and Hyperheuristics for Workforce Scheduling. In *Evolutionary Computation in Combinatorial Optimization*. Springer Berlin Heidelberg, 188–197. https://doi.org/10.1007/978-3-540-71615-0_17

Melissa Sánchez, Jorge M Cruz-Duarte, José carlos Ortíz-Bayliss, Hector Ceballos, Hugo Terashima-Marin, and Ivan Amaya. 2020. A systematic review of hyper-heuristics on combinatorial optimization problems. *IEEE Access* 8 (2020), 128068–128095.

T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

Jorge A Soria-Alcaraz, Gabriela Ochoa, Marco A Sotelo-Figeroa, and Edmund K Burke. 2017. A methodology for determining an effective subset of heuristics in selection hyper-heuristics. *European Journal of Operational Research* 260, 3 (2017), 972–983.

J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke. 2014. Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research* 238, 1 (2014), 77 – 86. https://doi.org/10.1016/j.ejor.2014.03.046

G. Tsoumakas, D. Vrakas, N. Bassiliades, and I. Vlahavas. 2004. Using the k Nearest Problems for Adaptive Multicriteria Planning. In *in Proceedings of the 3rd Hellenic Conference on Artificial Intelligence, SETN04*. Springer, 132–141. https://doi.org/10.1007/978-3-540-24674-9_15

Niki van Stein, Diederick Vermetten, Anna V Kononova, and Thomas Bäck. 2024. Explainable Benchmarking for Iterative Optimization Heuristics. *arXiv preprint arXiv:2401.17842* (2024).

J. Woodward, A. Parkes, and G. Ochoa. 2008. A Mathematical Formalization of Hyper-Heuristics. Workshop on Hyper-Heuristics Automating the Heuristic Design Process, presented at 10th International Conference on Parallel Problem Solving From Nature (PPSN-08) September 13-17, 2008 Technische University Dortmund, Germany.

W. B. Yates and E. C. Keedwell. 2017. Clustering of Hyper-heuristic Selections using the Smith-Waterman Algorithm for Offline Learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 119–120. http://dx.doi.org/10.1145/3067695.3076025

W. B. Yates and E. C. Keedwell. 2019. An Analysis of Heuristic Subsequences for Offline Hyper-heuristic Learning. *Journal of Heuristics* 25, 3 (2019), 399–430.

W. B. Yates and E. C. Keedwell. 2021. Offline Learning with a Selection Hyper-heuristic: An application to Water Distribution Network Optimisation. *Evolutionary Computation* 29, 2 (2021), 187–210.

Shuyan Zhang, Zhilei Ren, Cuixia Li, and Jifeng Xuan. 2020. A perturbation adaptive pursuit strategy based hyper-heuristic for multi-objective optimization problems. *Swarm and Evolutionary Computation* 54 (2020), 100647.

Yuchang Zhang, Ruibin Bai, Rong Qu, Chaofan Tu, and Jiahuan Jin. 2022. A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research* 300, 2 (2022), 418–427.

Fuqing Zhao, Shilu Di, Jie Cao, Jianxin Tang, et al. 2021a. A novel cooperative multi-stage hyper-heuristic for combination optimization problems. *Complex System Modeling and Simulation* 1, 2 (2021), 91–108.

M. Zhao, W-P. Lee, E. P. Garrison, and G. T. Marth. 2013. SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications. *PLOS ONE* 8, 12 (2013). https://doi.org/10.1371/journal.pone.0082138

Yanwei Zhao, Longlong Leng, and Chunmiao Zhang. 2021b. A novel framework of hyper-heuristic approach and its application in location-routing problem with simultaneous pickup and delivery. *Operational Research* 21 (2021), 1299–1332.

Marc-André Zöller, Waldemar Titov, Thomas Schlegel, and Marco F Huber. 2022. XAutoML: A Visual Analytics Tool for Establishing Trust in Automated Machine Learning. *arXiv preprint arXiv:2202.11954* (2022).