On-line Estimators for Ad-hoc Task Allocation

Extended Abstract

Elnaz Shafipour Yourdshahi[1], Matheus Aparecido do Carmo Alves[2],
Leandro Soriano Marcolino[1], Plamen Angelov[1]
[1] School of Computing and Communications, Lancaster University
[2] Institute of Mathematics and Computer Science (ICMC), University of São Paulo (USP)
elnaz.shafipour@lancaster.ac.uk, matheus.aparecido.alves@usp.br,
l.marcolino@lancaster.ac.uk, p.angelov@lancaster.ac.uk

ABSTRACT

It is essential for agents to work together with others to accomplish common missions without previous knowledge of the team-mates, a challenge known as ad-hoc teamwork. In these systems, an agent estimates the algorithm and parameters of others in an on-line manner, in order to decide its own actions for effective teamwork. Meanwhile, agents often must coordinate in a decentralised fashion to complete tasks that are displaced in an environment (e.g., in foraging, demining, rescue or fire control), where each member autonomously chooses which task to perform. By harnessing this knowledge, better estimation techniques would lead to better performance. Hence, we present *On-line Estimators for Ad-hoc Task Allocation*, a novel algorithm for team-mates' type and parameter estimation in decentralised task allocation. We ran experiments in the level-based foraging domain, where we obtain lower error in parameter and type estimation than previous approaches, and a significantly better performance in finishing all tasks.

ACM Reference Format:
Elnaz Shafipour Yourdshahi, Matheus Aparecido do Carmo Alves, Leandro Soriano Marcolino, Plamen Angelov. 2020. On-line Estimators for Ad-hoc Task Allocation. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 INTRODUCTION

In ad-hoc teamwork, agents collaborate to accomplish common tasks without pre-knowledge of each other, nor prior coordination or communication protocols. Hence, it is a beneficial model for solving issues in real-world domains, like rescue robots from different organisations which are urgently brought together to aid in a natural disaster – e.g., earthquakes. In these scenarios, designing coordination/communication protocols would take time, and resources. Avoiding such delays and funding usage could save lives.

Instead of learning models from scratch, it is common in the literature to assume a set of possible types [4, 5], reducing the problem to estimating the type of each agent. This approach is more applicable, as it does not require such a large number of observations, and can be more easily applied in an on-line manner. Types could be built based on previous experiences [6] or may be derived from the domain [1]. Additionally, having parameters for each type allowed more fine-grained models [2]. However, previous works were not specifically designed for decentralised task allocation, missing an opportunity to obtain better performances in this relevant scenario for multi-agent collaboration.

Note that individual agents do not need to share the same representation of the problem as decentralised task allocation, and run algorithms that explicitly "choose" tasks. They could be developed by different parties, and could use different paradigms. All we need are problems that can be modelled as decentralised task allocation for *our* ad-hoc agent. Similarly, a global allocation algorithm is unfeasible in our scenario: agents developed by others would not necessarily follow commands from a central entity, and we are not assuming any communication protocol.

Therefore, for better estimation of team-mates types and parameters in decentralised task allocation, we introduce our *novel algorithm* called *On-line Estimators for Ad-hoc Task Allocation* (OEATA). We run experiments in a collaborative foraging domain, where agents collect "heavy" boxes together. We obtain a lower error in parameter and type estimations in comparison with the state-of-the-art, leading to significantly better performance in task execution.

2 METHODOLOGY

We consider one agent ϕ, in the same environment as a set of agents Ω ($\phi \notin \Omega$). ϕ must maximise team performance, but it does not know how agents $\omega \in \Omega$ may behave at each state. As in previous works [2], we consider that agents in Ω can be defined by a type $\theta \in \Theta$, and by a vector of parameters $\mathbf{p}$, each in a fixed range. Estimating θ and $\mathbf{p}$ allows ϕ to estimate ω's behaviour, leading to better decision-making. Hence, we introduce OEATA for better parameter and type estimations in ad-hoc decentralised task allocation.

In OEATA, we have a set of *estimators* for each agent ω and each type θ ($\mathbf{E}^{\theta}_{\omega}$), which have a fixed size N. An *estimator* e is a tuple, $(\mathbf{p}_e, s_e, \tau_e, c_e, f_e)$: $\mathbf{p}_e$ is a parameter vector; s_e is the last *choose target state* (*choose target state* is the state s_τ when ω tries to choose a new task τ' after completing the task τ); τ_e is the task that ω would try to complete when having parameter $\mathbf{p}_e$ and type θ; c_e holds the number of times that e was successful in predicting ω's next task; f_e holds the number of *consecutive* failures. The s_e states are updated when another agent collects the predicted task τ_e. The *choose target state* of the agent is then estimated as the one stored in the *estimator* e with highest c_e across all sets $\mathbf{E}^{\theta}_{\omega}$. These will be used to compose the agent's history, when updating c_e.

All *estimators* e are initialised in the first step, and $\mathbf{p}_e$ of each e can be initialised with random values (e.g., from the uniform distribution). For all *estimators* e, s_e is set as the initial state of

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

the environment. Since each e has a certain type θ and a certain parameter vector $\mathbf{p}_e$, it allows ϕ to estimate ω's task decision process in the initial state. The estimated chosen task is assigned as τ_e.

When ω completes a task τ_ω, we start evaluating its *estimators*. For every e in $\mathbf{E}^\theta_\omega$, we check if τ_e is equal to τ_ω. If they are equal, we set c_e as the number of successes across the whole history so far, and set f_e to 0. Additionally, we store each p_i in $\mathbf{p}_e$ in a bag $\mathbf{b}_i$. Note that each position i of the parameter vector has a corresponding bag $\mathbf{b}_i$ to keep successful values, and there are sets of bags for each $\mathbf{E}^\theta_\omega$. If τ_e is not equal to τ_ω, we increase f_e and decrease c_e. If f_e is greater than a threshold ξ, we remove e. We also update the *choose target state* (s_e) and τ_e of all e.

We generate new *estimators* for each one removed, in order to again have $|\mathbf{E}^\theta_\omega| = N$. A proportion m of the new *estimators* are created by randomly sampling from the uniform distribution in the corresponding parameter's range; and the remaining proportion are created from the $\mathbf{E}^\theta_\omega$ bags. That is, for each position i of $\mathbf{p}_e$, we randomly sample a value from the corresponding bag $\mathbf{b}_i$. We then use the *average* $\mathbf{p}_e$ across all e in $\mathbf{E}^\theta_\omega$ as the current estimated parameter for ω, when assuming type θ.

We also estimate the probabilities $\mathrm{P}(\theta)_\omega$ of each agent ω having type θ. To do so, we use the success rate of all *estimators* of the corresponding type. For each θ, we first calculate: $k^\theta_\omega := \max(0, \sum_{e \in \mathbf{E}^\theta_\omega} c_e)$. These are then normalised as: $k'^\theta_\omega := \frac{k^\theta_\omega}{\sum_{\theta' \in \Theta} k^{\theta'}_\omega}$. Finally, we update the type estimation: $\mathrm{P}'(\theta)_\omega \propto k'^\theta_\omega \times \mathrm{P}(\theta)_\omega$, where $\mathrm{P}(\theta)_\omega$ is the previous estimation. In the first iteration, we need prior probabilities for $\mathrm{P}(\theta)_\omega$. These would normally be initialised with the uniform distribution, in the absence of previous information.

3 RESULTS

We evaluate OEATA in level-based foraging (Figure 1) [2, 3] . Agents collect items displaced in the environment (which corresponds to the "tasks"). Each item has a weight, and each agent has an (unknown) *skill-level*. If the sum of the skill-levels of the agents surrounding a target is greater than or equal the item's weight, the item is "loaded" by the team. The visibility region of each ω has an angle and a maximum radius, which are unknown. Hence, there are 3 parameters to be learnt: *Skill-level*, *Angle* and *Radius*. According to ω's type and parameters, its target item will be selected. We executed 100 runs for each experiment, and plot the average results and the confidence interval ($\rho = 0.01$). When we say that a result is significant, we mean statistically significant considering $\rho \leq 0.01$.

We compare our algorithm (*OEATA*) against two state-of-the-art parameter estimation approaches in ad-hoc teamwork: *AGA* and *ABU* [2]. Both approaches also sample sets of parameters (for a gradient ascent step or a *Bayesian* estimation), and we use the same set size as *estimator* sets (N). Additionally, we compare our approach against using *Partially Observable Monte-Carlo Planning (POMCP)* [7] for type and parameter estimations. In this case, we still consider that the agent is able to see the whole environment; however, agent type and parameters are not observable, and hence are estimated using POMCP's particle filter. We use $N \times |\Omega| \times |\Theta|$ particles, matching the total number of *estimators* in our approach (since we have N per agent, for each type).

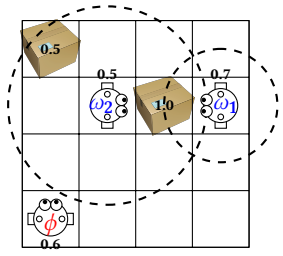

Figure 1: Level-based foraging domain.

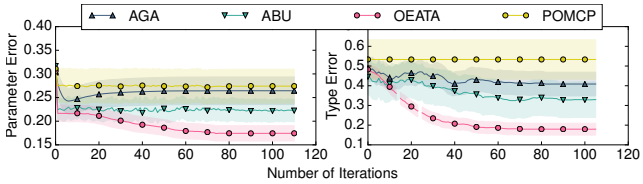

Figure 2: Parameter and type estimation errors for $|\Omega| = 5$.

OEATA used the following parameters: $N = 100$, $\xi = 2$, $m = 0.2$. Type and parameters of agents in Ω are chosen uniformly randomly, and the weight of each item is chosen uniformly randomly (between 0 and 1). Each scenario is also randomly generated, with 20 items. Agent ϕ's skill-level is fixed at 1, so every generated instance is solvable. We ran UCT-H (a variant of UCT [8]) for 100 iterations per time step, and maximum depth 100. We fix the scenario size as 20×20.

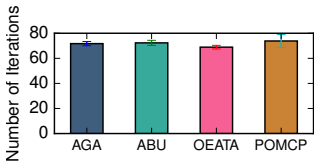

Figure 3: Performance for $|\Omega| = 5$.

We show examples of the parameter and type error for $|\Omega| = 5$ (Figure 2). We evaluate the mean absolute error for the parameters, and $1 - \mathrm{P}(\theta^*)$ for type (where θ^* is the true type); and we show here the average error across all parameters. As we can see, our parameter estimation error is consistently significantly lower than the other algorithms from the second iteration, and it monotonically decreases as the number of iterations increases. AGA, ABU, and POMCP, on the other hand, do not show any sign of converging to a low error as the number of iterations increases. We can also see that our type estimation becomes quickly better than the other algorithms, significantly overcoming them after a few iterations.

We demonstrate the performance of each estimation method in Figure 3, where we show the number of iterations to collect all items. As we can see, OEATA completed all tasks significantly faster than the other algorithms (in the case of POMCP, $\rho = 0.017$).

4 CONCLUSION

We study ad-hoc teamwork for decentralised task allocation. One ad-hoc agent learns about the decision-making algorithms of its team-mates, in order to better take decisions concerning overall team performance. By focusing on decentralised task allocation, we propose *On-line Estimators for Ad-hoc Task Allocation*, a novel algorithm which obtains lower error in parameter and type estimation than previous works, leading to better performance.

Acknowledgements: We thank AUSPIN and the School of Computing and Communications for their support. We also thank the High End Computing facility at Lancaster University.

REFERENCES

[1] S. Albrecht, J. Crandall, and S. Ramamoorthy. 2015. An empirical study on the practical impact of prior beliefs over policy types. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[2] S. Albrecht and P. Stone. 2017. Reasoning about Hypothetical Agent Behaviours and their Parameters. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'17)*.

[3] Stefano V. Albrecht and Subramanian Ramamoorthy. 2013. *A Game-Theoretic Model and Best-Response Learning Method for Ad Hoc Coordination in Multiagent Systems*. Technical Report. The University of Edinburgh.

[4] S. V. Albrecht and S. Ramamoorthy. 2016. Exploiting causality for selective belief filtering in dynamic Bayesian networks. *Journal of Artificial Intelligence Research* 55 (2016).

[5] Samuel Barrett, Peter Stone, and Sarit Kraus. 2011. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Vol. 2. 567–574.

[6] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. 2013. Teamwork with limited knowledge of teammates. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[7] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[8] E. S. Yourdshahi, T. Pinder, G. Dhawan, L. S. Marcolino, and P. Angelov. 2018. Towards Large Scale Ad-hoc Teamwork. In *Proceedings of the IEEE International Conference on Agents (ICA)*.