Learning Queuing Strategies in Human-Multi-Robot Interaction

Robotics Track

Masoume M. Raeissi
University of Verona
Verona, Italy
masoume.raeissi@univr.it

Alessandro Farinelli
University of Verona
Verona, Italy
alessandro.farinelli@univr.it

ACM Reference Format:
Masoume M. Raeissi and Alessandro Farinelli. 2018. Learning Queuing Strategies in Human-Multi-Robot Interaction. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

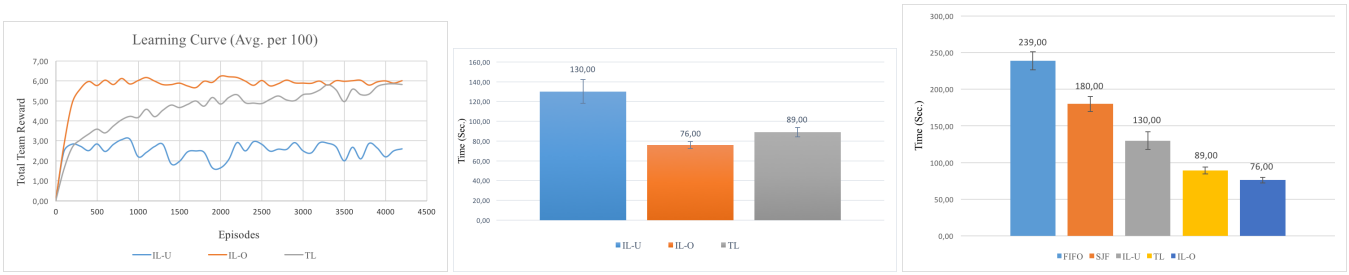
1 INTRODUCTION

We consider multi-robot applications, where a team of robots can ask for the intervention of a human operator to handle difficult situations. As the number of requests grows, team members will have to wait for the operator attention, hence the operator becomes a bottleneck for the system. Previous research try to enhance the performance of the system considering various queue disciplines (e.g. FIFO and SJF (shortest job first))[3, 6] or prioritizing the requests [9]. In both cases, the queue size may grow indefinitely as no robot will leave the queue before receiving the operator's attention. Our aim in this context is to make the robots learn cooperative strategies to decrease the time spent waiting for the operator. In particular, we consider the balking queue model [7], in which the agents can decide either to join the queue or balk. Such decisions are typically based on a threshold value, that is computed by assigning a generic reward associated with receiving the service and a cost for waiting in the queue to each agent. When applying this model to a robotic application, there is no clear indication on how such a threshold can be computed. More important, this model does not consider the cost of balking (i.e. the cost of a potential failure that the robot can suffer without human intervention). Our aim is to devise an approach that allows the robots to learn cooperative balking strategies to decrease the time spent waiting for the operator. In more detail, we formalize the problem as Decentralized Markov Decision Process (Dec-MDP)[1, 5] and provide a scalable state representation by adding the state of the queue as an extra feature to each robot's local observation. Solving Dec-MDP when the model of the environment is unknown (e.g. the arrival time of different events, the required time to resolve a request, etc.) is not trivial. Hence, we propose applying multi-agent reinforcement learning [2, 11] for our application and in general for similar human-multi-robot applications.

2 PROBLEM FORMULATION

We consider a water monitoring application[4, 10], where several autonomous boats are supervised by a human operator. A set of events can happen to the platforms, such events may affect the normal behavior of the platforms and hinder their performance. Each event has a different probability of failure. A central queue is provided to the operator and the boats, where the operator can select one request at a time (i.e., FIFO) and selects a specific sub-mission to resolve that request. Each sub-mission is a plan-specific recovery procedure. We assume that, whenever an event happens, the platform can detect the event. For example, the robot can perceive that its battery level is in a critical state, it must then decide whether to join the queue (i.e. sending the request and waiting for the operator) or balk (i.e., not sending the request). While this may be a significant challenge in some domains, this is not the focus of our work. The consequences of balking are problem specific. In our model, when a failure happens, the operator should spend more time to fix the problem, hence failure as a result of balking, increases the idle time of the system. As an example, consider event E_j happens to boat i, and it should select to join or balk. The decision of joining the queue will affect the future decisions of others, while choosing to balk may result in failure. We model the problem as a Dec-MDP. We formalize the state space with $S = S_1 \times S_2 \times ... \times S_N$ where N is the number of boats. The local state of each boat S_i is a tuple $\langle S_b, N_{tasks} \rangle$ where: N_{tasks} shows the number of remaining tasks for boat i. In this application domain, each task is a location that should be visited by a specific boat. S_b is the current internal state of boat i (e.g. whether it has a request, if it is waiting for the operator, etc.). More specifically $S_b \in \{E_j, \mathbf{W}aiting, \mathbf{F}ailed, \mathbf{A}utonomy\}$ where $j = 1, 2, ..., m$ is the cardinality of request/event types. In our model, $m = 3$ and $E_1 = BatteryRecharge$, $E_2 = TraversingDangerousArea$ and $E_3 = LosingConnection$ with probability of failure 0.9, 0.4 and 0.2 respectively. For example, the state tuple of a boat when it has 3 tasks to finish and the event *Battery Recharge* occurs, would be $s = \langle E_1, 3 \rangle$. A_i is the set of actions for boat i where $A_i \in \{Join, Balk\}$. Our proposal is then to train the robots, so that they can learn appropriate balking policies. We use Q-Learning[12], which is commonly used in robotic systems due to its effectiveness, as the basis learning approach for our learners. In general, there are two major approaches for learning in multi-robot scenarios [8, 13]. The first approach, team learning, uses a single learner to learn the behavior for the entire team. In contrast, the second approach uses multiple concurrent learners, usually one for each robot, where each learner tries to learn its behavior. The major problems with team learning approach are the explosion of the state space (i.e., it keeps the states of the entire team), and the centralization of the learning approach that needs to access the states of all team members. For example, in our domain for 5 boats with the above state representation, the state space will include more than one million states, hence requiring a prohibitive long time to estimate the optimal strategies for each state and action permutations. On the other hand, the main advantage of independent learners is that, our domain can be decomposed

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.



(a) Team accumulated reward (b) Total waiting time (c) Total waiting time

Figure 1: (a) the team accumulated reward in each episode of the learning phase. (b) the team performance (together with the standard error of the means) for three learning models. (c) comparison between balking models to non-balking models.

into subproblems (e.g. each boat holds its own state space) and each subproblem can be solved by one boat. In general, two main challenges arise in independent learning: credit assignment and non-stationary dynamics of the environment [8]. However, our application scenario has some special properties, that can be exploited to design a tractable model. In particular, the action selection at each step (i.e. when an event happens) only requires one agent to select either to join or balk. Hence, the reward can go directly to that agent. However, when each boat considers only its local state without knowing the state of the queue, finding the optimal behavior for the team may become impossible, or the model may compute lower quality solutions. Therefore, we add the state of the queue to the local state of each boat, and then we use independent learners approach. To sum up, we consider three possible models: **Team Learner (TL)**, where a team learner has access to the joint state of all robots which is $S = S_1 \times S_2 \times \ldots \times S_N$. When an event happens to a boat, the action $\langle Join, Balk \rangle$ for the corresponding boat will be selected and the state of the system will be updated. The update will only change the part of the state related to the corresponding boat. **Independent Learners - Unobservable Queue (IL-U)**, where an independent learner is used for each boat. Each boat observes only its local state $S_i = \langle S_b, N_{tasks} \rangle$. In this model, each boat updates its local Q-values interacting with the system and receiving the reward. **Independent Learners - Observable Queue (IL-O)**, where each boat in addition to observing its local state, has access to queue size (i.e. number of waiting boats): $S_i = \langle S_b, N_{tasks}, S_q \rangle$. The three models are different in their state representation, while the reward structure is the same for all of them: **(i)** $R(S_t = S_i, A_t = Join) = R_S - (N_q \bar{\mu} + t_{serv})$. **(ii)**$R(S_t = S_i, A_t = Balk) = R_F(\frac{\bar{\mu}}{\bar{\lambda}}) + N_q$; *if* $S_{t+1} = F$. **(iii)** $R(S_t = S_i, A_t = Balk) = R_T$; *if* $S_{t+1} = A$. $\bar{\mu}$ and $\bar{\lambda}$ are average service time and events arrival rate respectively. N_q is the number of boats waiting, and t_{serv} is the average time needed to resolve the request. $R_S = 1$, $R_F = -2$ and $R_T = 0.3$ are application specific parameters that must be tuned empirically.

3 EXPERIMENTAL EVALUATION

The learning phase starts by defining a list of locations to be visited, and assigning those to boats. We consider 30 locations and 5 boats. Events are generated within an exponential distribution with parameter $\lambda = 0.25$. The operator's speed, for resolving a request is selected from an exponential distribution with parameter $\mu = 0.27$. We use ϵ_greedy method for action selection with $\epsilon = 0.1$. Our algorithm uses the learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.9$ throughout the experiments, which were tuned empirically. Each episode of the learning phase starts with all boats in their Autonomy state (i.e. they do not need the attention of the operator), then with rate λ an event may happen to one boat. We used a realistic estimation for parameters λ and μ based on some experience on the total mission time, number of boats and number of locations. These numbers define well the type of scenarios we are interested in, where boats can operate most of the time in autonomy, but frequently need user's intervention. Figure 1(a) shows the team rewards of each model, *TL*, *IL-U* and *IL-O*, during the learning phase. The oscillation in the reward is because, the robots learn their policies by trying new, potentially sub-optimal actions. As expected, the convergence rate of *IL-O* is much faster than *TL*, while they both reach a similar team reward. This is due to the larger state space of *TL* which needs more iterations to estimate the value for each state and action. After the learning phase, we run 30 simulation executing the policy learned previously. We use the same values for λ and μ as used during the learning phase. Figure 1(b) demonstrates the idle time for each learning models. Next, we compare the behavior of queues with and without balking property. For FIFO and SJF, we use the same event rate λ and service rate μ. In these two queuing models, boats always join the queue regardless of their request types and the queue size. Figure 1(c) shows that, FIFO without balking has the worst performance, since boats wait for the operator until he/she becomes available. In contrast, *IL-O* outperforms all other models. In general, the results in Figure 1(c) indicate that, using balking models significantly decreases the idle time of the team even though, some events may result in failures. This is acceptable in our domain, since the penalties for failures are not critical, but only result in a finite increase of time.

ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 689341. This work reflects only the authors' view and the EASME is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[2] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. 2015. Evolutionary Dynamics of Multi-Agent Learning: A Survey. *J. Artif. Intell. Res.(JAIR)* 53 (2015), 659–697.

[3] Shih Yi Chien, Michael Lewis, Siddharth Mehrotra, Nathan Brooks, and Katia P. Sycara. 2012. Scheduling operator attention for Multi-Robot Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 473–479. https://doi.org/10.1109/IROS.2012.6386019

[4] Alessandro Farinelli, Masoume M. Raeissi, Nicolo' Marchi, Nathan Brooks, and Paul Scerri. 2017. Interacting with Team Oriented Plans in Multi-robot Systems. *Autonomous Agents and Multi-Agent Systems* 31, 2 (March 2017), 332–361. https://doi.org/10.1007/s10458-016-9344-6

[5] Claudia V. Goldman and Shlomo Zilberstein. 2003. Optimizing Information Exchange in Cooperative Multi-agent Systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03)*. ACM, New York, NY, USA, 137–144. https://doi.org/10.1145/860575.860598

[6] Michael Lewis, Shi-Yi Chien, Siddarth Mehortra, Nilanjan Chakraborty, and Katia Sycara. 2014. *Task Switching and Single vs. Multiple Alarms for Supervisory Control of Multiple Robots*. Springer International Publishing, Cham, 499–510. https://doi.org/10.1007/978-3-319-07515-0_50

[7] P. Naor. 1969. The Regulation of Queue Size by Levying Tolls. *Econometrica* 37, 1 (1969), 15–24. http://www.jstor.org/stable/1909200

[8] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

[9] Ariel Rosenfeld, Noa Agmon, Oleg Maksimov, Amos Azaria, and Sarit Kraus. 2015. Intelligent Agent Supporting Human-multi-robot Team Collaboration. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1902–1908. http://dl.acm.org/citation.cfm?id=2832415.2832513

[10] Paul Scerri, Balajee Kannan, Pras Velagapudi, Kate Macarthur, Peter Stone, Matt Taylor, John Dolan, Alessandro Farinelli, Archie Chapman, Bernadine Dias, and George Kantor. 2012. *Flood Disaster Mitigation: A Real-World Challenge Problem for Multi-agent Unmanned Surface Vehicles*. Springer Berlin Heidelberg, Berlin, Heidelberg, 252–269. https://doi.org/10.1007/978-3-642-27216-5_16

[11] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

[12] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (01 May 1992), 279–292.

[13] Ping Xuan and Victor Lesser. 2002. Multi-agent Policies: From Centralized Ones to Decentralized Ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3 (AAMAS '02)*. ACM, New York, NY, USA, 1098–1105.