Multiagent Task Allocation and Planning with Multi-Objective Requirements

Extended Abstract

Thomas Robinson
University of Wollongong
Wollongong, Australia
tmr463@uowmail.edu.au

Guoxin Su
University of Wollongong
Wollongong, Australia
guoxin@uow.edu.au

Minjie Zhang
University of Wollongong
Wollongong, Australia
minjie@uow.edu.au

ABSTRACT

In service robot applications, planning is often integrated with task allocation. Linear Temporal Logic (LTL) as an expressive high-level formalism is widely used for task specification, and allows for formalised restrictions on temporal sequences of tasks. In multiagent planning, a Multi-Objective Markov Decision Process extends the standard model with vector rewards capturing possibly conflicting planning objectives. Such objectives include the success rates of accomplishing individual tasks, and the cost budgets for individual agents. In this paper, we consider the problem of concurrently allocating LTL task sequences to a team of agents and calculating optimal task schedulers simultaneously, satisfying cost and probability thresholds. We reduce this problem to multi-objective scheduler synthesis for a team MDP structure, whose size is linear in the number of agents. Our preliminary experiment demonstrates the scalability of our approach.

KEYWORDS

Multiagent Planning; Multi-Objective Planning; Task Allocation

ACM Reference Format:
Thomas Robinson, Guoxin Su, and Minjie Zhang. 2021. Multiagent Task Allocation and Planning with Multi-Objective Requirements: Extended Abstract. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

Traditionally, multiagent planning aims at computing a solution (i.e., scheduler) for agents to achieve a pre-defined objective. In many real-world applications of service robots, a set of possibly heterogeneous tasks are assigned to a team of robots to accomplish. Furthermore, those service robots may be constrained by cost budgets. For example, a set of tasks requiring different numbers of consecutive sprints are assigned to a team of robots, which have different success rates of sprints and cause different costs (i.e., energy consumption or maintenance cost). For a multi-robot system, planning is often integrated with allocating tasks to the robots, coined as a (simultaneous) *task allocation and planning* (TAP) problem in [24]. Separating allocation from planning reduces the complexity of a TAP problem but may result in a sub-optimal solution.

In this paper, we consider the TAP problem for multiagent systems with respect to multi-objective requirements. Following [24], we adopt a fragment of LTL, called co-safe LTL [25], to specify the tasks. As we are interested in stochastic planning, our approach is close to [5], which extends the TAP framework of [24] to the uncertainty setting, and in particular, employs MDPs as agent models. However, while [5] solves the TAP problem with respect to an overall probability requirement, we aim to deal with a *Multi-Objective version of the TAP problem (MOTAP)* which involves both probability and cost requirements.

In this paper we formulate the MOTAP problem as a randomised allocation of tasks to agents and reduce the problem to a multi-objective scheduler synthesis problem for a team MDP structure. The overall aim is to provide an approach to solving a multi-objective task allocation and planning problem, scaling linearly in the number of agents.

2 APPROACH

Tasks as Automata: LTL formulas φ over a finite set AP of atomic propositions are defined using the following grammar:

$$\varphi ::= \top \mid \mathsf{a} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{X}\,\varphi \mid \varphi_1\,\mathsf{U}\,\varphi_2 \tag{1}$$

where $\mathsf{a} \in AP$. The operators X and U stand for "next" and "until", respectively.

It is natural to consider task allocation which completes in finite time, especially in the circumstance where we consider allocating sequences of tasks to a single agent. Task formulation which completes in finite time requires a subset of LTL formulas which are guaranteed to have a finite fragment. We then only want to consider the *co-safe* language $\mathcal{L}^c = \Sigma^\omega \setminus \mathcal{L}_{safe}$, where if $x \in \Sigma^*$ and an accepting prefix, and $y \in \Sigma^\omega$ then $x.y \in \mathcal{L}$. It has been shown that any LTL formula containing only the temporal operators X (*next*), U (*until*), an F (*eventually*) in positive normal form (PNF) always results in a co-safe property satisfying $\mathcal{L}^c$ [12]. Further, because the infinite part of a co-safe LTL formula does not affect the outcome of any sequence [12], we may define a non-deterministic finite automaton (NFA) which is always accepting for the finite sequence [2]. It is well known that deterministic finite automata (DFA) and NFA are equally expressive so then we may construct a DFA, $\mathcal{A}_j = (Q_j, q_{j,0}, Q_{j,F}, \Sigma, \delta)$ for any co-safe LTL task formula.

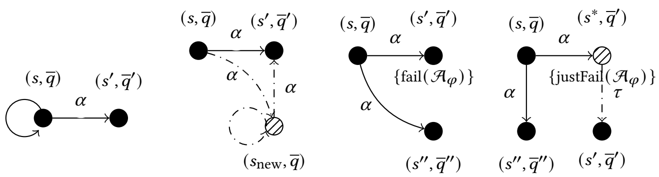
Agent models: An agent with some expectation of uncertainty in the outcome of its actions is represented by a labelled Markov decision process (MDP)[1] defined as a tuple $\mathcal{M} = (S, s_0, A, P, L)$ where S is a finite state space, $s_0 \in S$ is an initial state, A is as set of actions, $P : S \times A \times S \mapsto [0, 1]$ is a transition function, $L : S \mapsto \Sigma$ is a labelling function. We want to verify that the

[1]See [2] for more details on model checking MDPs

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Table 1: Model size and runtime comparison between the team MDP and MAMDP in Experiment 1, where t - runtime in seconds, $|S|$ - state space, $|P|$ - number of transitions, $|S|'$ - reachable states, $|P|$ - number of reachable transitions.

#Agents	Team MDP			MAMDP				
	t	$\|S\|$	$\|P\|$	t	$\|S\|$	$\|P\|$	$\|S\|'$	$\|P\|'$
3	9	4320	9504	9	26091	259853	1000	9000
4	12	7488	21488	77	260901	3.3×10^6	10000	120000
5	17	9360	34992	627	2.1×10^6	2.9×10^7	100000	1.5×10^6
6	23	11232	50976	7585	2×10^7	3.6×10^8	1×10^6	1.8×10^7
7	30	13104	69120	-	-	-	1×10^7	2.1×10^8



(a) A pre-modified self loop **(b)** A modified self loop **(c)** A pre-modified failure **(d)** A modified failure

Figure 1: Modification diagram for self loops and failures to accurately evaluate task rewards.

traces generated on a labelled MDP is exactly those accepted by the collection of task DFA. To do this we can iteratively construct a *product MDP* $\mathcal{M}_i \otimes \mathcal{A} = (\mathcal{M}_i \otimes \mathcal{A}_1) \otimes \mathcal{A}_2) \dots) \otimes \mathcal{A}_m$ for agent i using the collection of task DFA $\{\mathcal{A}_1, \dots, \mathcal{A}_m\}$ according to [2] for which the resulting state space is $S_i \times \overline{Q}$. A standard product MDP is unsuitable for optimising task allocation and requires a number of modifying steps, the resulting structure is distinguished as $\mathcal{M}_i \underline{\otimes} \mathcal{A} = (S_i \times \overline{Q}, (s_{i,0}, \overline{q}_0), A_i, P_i, L_i)$ for agent i. The goal of an agent is to minimise the paths of an allocated task leading to failure and checking that the failure probability is within some acceptable threshold. Therefore, it is natural that once a task has been initiated by an agent it is carried out to completion. To account for this we make the following modification to self-loops, for each $\alpha \in A'(s, \overline{q})$ redirect the self-loop of $(s, \overline{q})$ under the action α to a pair $(s_{\text{new}}, \overline{q})$ as long as q_i is not accepted or rejected, demonstrate in Figures (1a, 1b). Further, rewards (penalties) should only be assigned for task failure when a task has reached a failure state for the first time. We add another new state s^* if there is a non-zero probability of transitioning from a non-failure state to one that satisfies failure for task j, shown in Figures (1c, 1d). States in the local product for which some combination of tasks are initial or completed/rejected are labelled Stoppable$(\mathcal{A}_j)$.

Rewards Structures: A rewards function maps state-action pairs to an n(agents) + m(tasks) rewards vector $\overline{\rho}(s, a) = (c_1, ..., c_n, t_1, ..., t_m)$ where $c_i = \rho((s, \overline{q}), a)$ inherited from the agent MDP when a task is *initial* or *in-progress* for agent i and 0 everywhere else. The elements $t_i \in \overline{\rho}((s, \overline{q}), a)$ are task rewards derived from Equation 2 below. Computing optimal schedulers relies on optimising convex combinations of expected total rewards using $\overline{\rho}((s, \overline{q}), a)$.

$$\overline{\rho}_{j+n}((s, \overline{q}), a) = \begin{cases} 1 & \text{if } (s, \overline{q}) \models \text{justFail}(\mathcal{A}_j) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Problem Definition: A collection of tasks is a valid *mission* if three conditions hold: (i) only one task is active at any given time, (ii) there is only one action relevant to a task progress in any reachable state, (iii) after a task is completed the MDP returns back to its initial state. Then given a set of agents and rewards structures, a multi-objective task allocation and planning (MOTAP) problem is to find a randomised allocation function which satisfies task completion probability thresholds and each agents cost constraint. A randomised allocation function is derived from the pareto optimal schedulers, and optimally allocates tasks to agents in a memoryless way. Therefore, tasks can be concurrently distributed to agents, who carry out tasks according to an optimal plan. To solve the MOTAP problem we require a team MDP, $M_I = (S_I, s_{I,0}, A_I, P_I, L_I)$ constructed in a similar way to [5], in the sense that a team MDP is a collection of local products with virtual switch transitions $b \in A_I$. In our team MDP, switch transitions exist between any state labelled with Stoppable$(\mathcal{A}_j)$ to an initial state of the $i+1 \in I$ agent, $P_I((s, \overline{q}), b, (s_{i+1,0}, \overline{q}))$.

3 EXPERIMENTS

Our approach was evaluated for model size, number of states $|S|$ and $|P|$ transitions, and runtime t. The aim of the experiment was to determine how our approach scaled in the number of agents compared to a traditional implementation of an asynchronous multiagent MDP (MAMDP) in PRISM [13] for the same set of agents and LTL formulation.

The outcome of the experiment can be observed in Table 1. When scaling in the number of agents, there was a significant difference in the rate of change between team MDP and the MAMDP, with the former having a much lower model checking time and a much smaller model size. Scaling the multiagent model to 7 agents caused PRISM to experience an out of memory exception. As the number of agents increased, the total model checking time for the team MDP grew linearly as expected, while that for MAMDP grew exponentially. For the team MDP there is a slight divergence between state-space growth and number of transitions (which is not observed in the MAMDP multi-objective model checking). That is due to the additional switch transitions handing over control of tasks to the next agent, which increased the number of transitions, but that increment was small compared with the size of state space. Our findings in this experiment support the claim that MOTAP is more suitable for handling larger multiagent systems than conventional models.

4 CONCLUSION AND FUTURE WORK

This paper proposed an efficient approach for allocating tasks, specified as co-safe LTL, that synthesised randomised schedulers for MDP agents meeting multi-objective requirements involving both cost and probability thresholds. Several future directions of the current work exist. While the team MDP allows decoupling of agents, there is possibly a further extension to decouple tasks. Exploration of the framework to remove the condition of an MDP returning to its initial state after task completion. Last, incorporating *multi-objective reinforcement learning* [21] into our framework to perform task allocation and planning in the model-free setting.

REFERENCES

[1] Christel Baier, Holger Hermanns, and Joost-Pieter Katoen. 2019. The 10,000 facets of MDP model checking. In *Computing and Software Science*. Springer, 420–451.

[2] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. The MIT Press, Cambridge, Mass. OCLC: ocn171152628.

[3] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A Henzinger. 2006. Markov decision processes with multiple objectives. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 325–336.

[4] Peng Dai, Daniel S Weld, Judy Goldsmith, et al. 2011. Topological value iteration algorithms. *Journal of Artificial Intelligence Research* 42 (2011), 181–209.

[5] Fatma Faruq, David Parker, Bruno Laccrda, and Nick Hawes. 2018. Simultaneous Task Allocation and Planning Under Uncertainty. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Madrid, 3559–3564. https://doi.org/10.1109/IROS.2018.8594404

[6] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. 2011. Quantitative multi-objective verification for probabilistic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 112–127.

[7] Vojtěch Forejt, Marta Kwiatkowska, and David Parker (Eds.). 2011. Automatic Verification Techniques for Probabilistic Systems. *Formal Methods for Eternal Networks* 6659 (2011), 60–120. https://doi.org/10.1007/978-3-642-21455-4

[8] Vojtěch Forejt, Marta Kwiatkowska, and David Parker. 2012. Pareto curves for probabilistic model checking. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 317–332.

[9] Meng Guo and Dimos V Dimarogonas. 2015. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research* 34, 2 (2015), 218–235.

[10] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. 2017. Multi-objective robust strategy synthesis for interval Markov decision processes. In *International Conference on Quantitative Evaluation of Systems*. Springer, 207–223.

[11] Marechal Juin and Abdel-Illah Mouaddib. 2006. Collective multi-objective planning. In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*. IEEE, 43–48.

[12] Orna Kupferman and Moshe Y Vardi. 2001. Model Checking of Safety Properties. *Formal methods in system design* 19, 3 (2001), 291–314.

[13] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.

[14] Marta Kwiatkowska, David Parker, and Hongyang Qu. 2011. Incremental quantitative verification for Markov decision processes. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 359–370.

[15] Matt Luckcuck, Marie Farrell, Louise A Dennis, Clare Dixon, and Michael Fisher. 2019. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–41.

[16] Erion Plaku and Sertac Karaman. 2016. Motion planning with temporal-logic specifications: Progress and challenges. *AI communications* 29, 1 (2016), 151–162.

[17] Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 46–57.

[18] Amir Pnueli and Roni Rosner. 1989. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 179–190.

[19] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[20] Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. 2020. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 10.

[21] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[22] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. 2014. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1525–1532.

[23] Philipp Schillinger, Mathias Bürger, and Dimos V. Dimarogonas. 2018. Decomposition of Finite LTL Specifications for Efficient Multi-agent Planning. In *Distributed Autonomous Robotic Systems*, Roderich Groß, Andreas Kolling, Spring Berman, Emilio Frazzoli, Alcherio Martinoli, Fumitoshi Matsuno, and Melvin Gauci (Eds.). Vol. 6. Springer International Publishing, Cham, 253–267. https://doi.org/10.1007/978-3-319-73008-0_18 Series Title: Springer Proceedings in Advanced Robotics.

[24] Philipp Schillinger, Mathias Bürger, and Dimos V. Dimarogonas. 2018. Simultaneous Task Allocation and Planning for Temporal Logic Goals in Heterogeneous Multi-Robot Systems. *The International Journal of Robotics Research* 37, 7 (June 2018), 818–838. https://doi.org/10.1177/0278364918774135

[25] Moshe Y Vardi and Pierre Wolper. 1994. Reasoning about infinite computations. *Information and computation* 115, 1 (1994), 1–37.