Multi-Agent Reinforcement Learning for Schedule-Constrained Automation

Kareem Eissa, Rayal Prasad, Ankur Kapoor
Siemens Healthineers
{kareem.abdelrahman, rayal.prasad, ankur.kapoor}@siemens-healthineers.com

Abstract

In modern automation settings, jobs are processed across numerous machines and are characterized by strong inter-task dependencies while adhering to limited equipment availability. When accounting for transportation of jobs between machines, this gives rise to a complex multi-agent routing problem with intricate operational limitations. Existing Multi-Agent Path Finding (MAPF) algorithms used for routing jobs already consider some aspects such as robustness, uncertainty, and plan execution. In this paper, we propose MAPF-SC – a lifelong variant of MAPF that incorporates scheduling constraints for a continuous stream of tasks. We propose to solve MAPF-SC utilizing a Multi-Agent Reinforcement Learning (MARL) formulation with temporal and team reward. We investigate the effects of temporal and topological variations of various automation scenarios on the performance of our method.

1 Introduction

State-of-the-art automation settings involve multiple interacting components that require coordinated actions. These scenarios can be conceptualized as multiple jobs simultaneously being transported to successions of machines for task processing. Whether it is logistics, manufacturing, or laboratory and healthcare automation, the Multi-Agent Path Finding (MAPF) formulation can be used to address important aspects of these target domains such as collision avoidance and travel time minimization to ensure optimal and safe motion between machines. Recent works have shown significant progress in different variants of MAPF [Li et al. 2021; Shahar et al. 2021; Chudý et al. 2021].

One key assumption prevalent across MAPF variants is instantaneous task processing which is atypical in many real-world situations. Even in the popular warehouse setting, a robot agent delivering a payload would certainly require a non-instantaneous amount of time at the processing station. This discrepancy is amplified in domains such factory automation, where tasks demand specific non-trivial processing as a part of the automation pipeline. The automation workflow may also be subject to restricted space due to costs or physical constraints such as ventilation, power outlets, structural constraints, etc. Machines may further require additional time between tasks for operations such as component switching or decontamination. Finally, a job would have a specific sequence of operations to undergo, leading to strong precedence constraints. Under the classical MAPF formulation, agent congestions will arise as they fail to capture the precedence constraints and queuing effects induced by the intricacies of the automation pipeline.

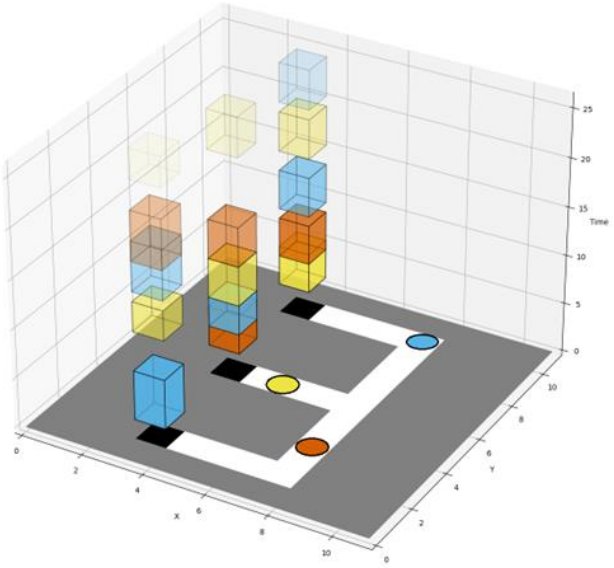

Figure 1: Example MAPF-SC instance. The bottom plane encodes the spatial information. Agents (circles) need to go to goals (black squares) according to schedules (colored rectangles). The elevation of the rectangles represents the earliest arrival time, and its height represents the duration of the time-window.

In this work, we formulate the MAPF-SC problem with execution schedules and precedence constraints. Such constraints address the challenges of automation domains characterized by earliest arrival times, deadlines, non-instantaneous task processing times, and inter-task precedence constraints. The schedule is enforced by the automation pipeline capabilities i.e., machine operating characteristics. To handle this, we model Multi-Agent Path Finding with Scheduling

Constraints (MAPF-SC) as a sequential decision-making problem that builds upon recent works [Sartoretti et al. 2019; Damani et al. 2021; van Knippenberg et al. 2021] showing promising results in adapting to uncertain and dynamical environments in a scalable decentralized manner.

2 Related Work

Classical MAPF is concerned with finding collision-free paths for a group of agents from their start locations to their goal locations. There are many lines of work on solving MAPF [Surynek et al. 2019; Hoenig et al. 2016]. The related literature we focus on involves MAPF variants that factor in time, as well as Reinforcement Learning approaches to the problem.

2.1 MAPF with Time

Several works have studied temporal aspects of MAPF. Wang et al. [Wang and Chen 2022] focus on minimizing the violation of task specific due times. More recently, Gao et al. [Gao et al. 2023] focus on the maximizing the average customer satisfaction proportional to the degree of deadline violation. Another line work [Ma et al. 2018; Zhang et al. 2022] focuses on the problem of maximizing the number of agents that reach their goals within a global deadline or satisfying precedence-constraints among goal sequences. The Flatland environment [Mohanty et al. 2020] simulates the railway setting as a multi-agent problem with the additional motion and schedule constraints. However, trains are allowed to park off the map before their departure and after their arrival, limiting the complexity of the planning problem.

2.2 Reinforcement Learning for MAPF

Early works [Sartoretti et al. 2019; Damani et al. 2021] model MAPF as a MARL problem and utilize expert training to train decentralized policies with a mixture of imitation learning and reinforcement learning. Another line of work focus on solving MAPF with arrival and deadline constraints using Reinforcement Learning [Knippenberg et al. 2021] but are limited to instantaneous task execution and may remove agents outside their scheduled times similar to Flatland.

3 Problem Formulation

The MAPF-SC problem is defined by an undirected graph $G = (V, E)$, a set of m agents $\{a_1, \dots, a_m\}$, and a schedule S. An example instance is illustrated in figure 1. These agents represent the jobs within the schedule, each having a sequence of associated tasks. The set of vertices V corresponds to locations on the graph and the set of edges E represents motion constraints for each vertex. At each step, an agent can either move to one of the unoccupied neighboring vertices or wait at its current vertex. Each agent i is assigned a sequence of goals corresponding to vertices $\{v_i^1, v_i^2 \dots\}$ according to the schedule. The schedule represents time-windows of goal j for agent i consisting of Earliest Arrival Time $A_t(i,j)$, Deadline $D_t(i,j)$, and Goal Processing Time $P_t(i,j)$. An agent may arrive at its goal prior to $A_t(i,j)$, but the task will not be processed until the scheduled time $A_t(i,j)$. $D_t(i,j)$ indicates the latest time that an agent is allowed to reach its goal for processing. $P_t(i,j)$ represents the task processing time during which the agent is not allowed to move from the goal position. Figure 2 demonstrates relationship between these temporal components. Scheduling constraints may also involve precedence among tasks assigned to the same machine where the order of arriving agents needs to be preserved. As in typical job shop automation workflows, the precedence constraints have a higher priority than the time-window constraints i.e., even if some agents are delayed past their deadlines, the order in which the machine processes them remains the same. These situations arise when machines require some postprocessing (e.g., replenish resource) after completing a task, leading to a lag in the execution of the schedule. Finally, agents are always present on the graph from the beginning and do not disappear after completing their goals - the persistence of all agents on a space-constrained track is a key challenge we aim to address.

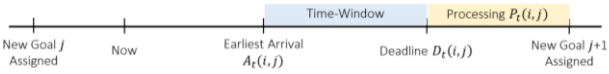

Figure 2: Different aspects of scheduling constraints.

4 Method

We model agent interactions with the environment as a Partially Observable Markov Decision Process (POMDP) (S, O, A, P, R, γ) where S is the set of environment states, O is the set of partial observations, A is the set of actions, P is the transition probabilities, R is the reward function, and γ is the discount factor. We aim to learn a decentralized policy that can be deployed to different numbers of agents.

4.1 Observations

The observation space consists of two main components. The first component is the agent-centered Local Field of View (FOV) which builds upon earlier related works [Sartoretti et al. 2019; Damani et al. 2021]. The FOV consists of channels denoting obstacles, the presence of other agents, and other agents' target locations. An additional channel provides the distance from each cell in the FOV to the agent's goal. In our experiments, we use a 7x7x5 FOV.

The second component encodes the agent's task as vectors. The spatial context of the task is represented through a three-dimensional vector, $\boldsymbol{G_s}$, as the difference in x-coordinates, difference in y-coordinates, and the magnitude of the vector from the agent's current location to its goal. The context is captured in another three-dimensional vector, $\boldsymbol{\tau_t}$, that provides the remaining time until the earliest arrival time A_t, the duration until the deadline D_t, and a binary delay indicator if the deadline D_t has passed. The dimensions of this…

4.2 Rewards

Contrary to classical MAPF, we do not assign a motion penalty to minimize the travel distance. Our intuition is that an

agent may have surplus time until its earliest arrival time A_t. Such an agent may potentially take a longer path to allow other agents to pass through. Taking this into consideration, minimizing only the travel distance may be detrimental to the throughput due to induced congestion. In our setup, each agent receives a positive reward when it reaches its goal within the scheduled time-window and a negative reward for each time step it occupies its goal before its earliest arrival time A_t. Similarly, agents receive a negative reward for each time step it has not reached its goal past the deadline D_t to incentivize arriving on time. However, agents may fail to meet their schedules on-time especially early on in training. Rather than terminating the episode, we permit soft deadline violations to allow the schedule's continued execution. All precedence constraints of the schedule remain enforced through the environment, which results in a queueing system for agents that have overlapping goal locations.

4.3 Architecture

We process the local FOV with a 3-layer convolutional neural network, each layer followed by a max pooling operation (except for the last one which is followed by a global average pooling) to produce a FOV representation vector. The task vectors (location and time) are then subsequently fed through fully connected layers to produce a combined task representation vector. Both the FOV and task vectors are then passed through a recurrent module to incorporate information about past states as a mitigation strategy for partial observability problems. The model is optimized with Proximal Policy Optimization which has shown great results on cooperative multi-agent settings [Yu et al. 2022].

5 Experiments

We provide a comprehensive overview of the layout design's attributes and its influence on input spatial characteristics. Simultaneously, we adjust the time-windows' parameters to induce variations in the resulting schedule slack.

5.1 Layouts

While scaling to a larger number of agents within the same fixed layout is an important aspect of MAPF, we argue that it does not hold the same value in the time-constrained scenario where the throughput of the entire system is limited by the schedule capacity itself. In most automation settings, the tasks are finite and are rate-limited by the machines that process them. Adding more agents will simply lead to them wait idle and have sufficient time to navigate the track. This means that as the number of agents increases in MAPF-SC, the bottleneck becomes then the schedule itself rather than the path finding algorithm.

To investigate these effects, we focus on more restrictive layout designs as shown in the *Layout* column in table 1. These capture variations in restricted tracks connected by single lane corridors to capture situations with limited spatial resources e.g., repurposing an existing automation plant. We vary the layouts across three design dimensions:

- *Redundancy*: Lanes are connected to remove dead ends and provide continuous coverage of the entire layout.
- *Size*: The number of machines and corresponding lanes are varied to create small (3) and large (6).

The agent count is adjusted proportionally and set to the number of machines + 2.

- *Padding*: Decision regions where agents have more than two degrees of freedom are padded with additional vertices.

5.2 Schedules

To replicate the temporal intricacies found in automation domains, we introduce variations in schedule distributions across two dimensions, similar to the approach in the Flatland environment [Mohanty et al. 2020; Laurent et al. 2021]:

- *Single-agent shortest distance* (A* factor) which is the minimum distance traveled by an agent assuming there are no other agents in the environment.
- *Multi-agent congestion factor* where we add the average of all single agent shortest distances as a congestion estimate.

These dimensions represent scheduling constraints that influence the earliest arrival time for each agent providing minimal necessary conditions for the schedule to be feasible. Additionally, we vary the size of the time-windows from earliest arrival time to deadline, and the task processing runtime before completion. We investigate two distributions of schedules we use:

- *Tight*: A* factor is sampled uniformly from [1, 2] and task durations are sampled uniformly from [4, 11].
- *Relaxed*: A* factor is sampled uniformly from [1.5, 3] and task durations are sampled uniformly from [5, 25].

We generate schedules specific to each layout using OR-Tools [Perron and Furnon 2023] where the solver optimizes for a randomly sampled list of agent tasks.

6 Results and Discussion

Through the experiments, we note that the model tends to complete more goals on-time on a relaxed schedule versus a tight one. Intuitively, this reflects a less constrained optimization problem since agents have less restricted time-windows to reach their goals. Table 1 summarizes the results for different spatial and temporal configurations.

In the closed-loop layouts (table 1, rows 1-4), we find that the agents learn to maintain a state of steady flow by circling the track. Upon nearing their earliest arrival times, we observe deviation from this behavior as agents move directly towards their goals. This continuous flow behavior globally reduces congestion. In the padded layouts with dead end lanes (table1, rows 5-6), agents learn maneuvering behaviors through the padded sections surrounding the decision regions. Agents also tend to use these padding vertices as "parking" to wait for their scheduled goals without blocking other agents.

In general, we find the most restrictive instances (table 1, rows 7-8) to be less stable in training and the RL policy often

Layout	Schedule	*On-Time %*		*Late %*	
		Mean	***Std***	***Mean***	***Std***
Layouts with redundancy					
 (padded)	Relaxed	97.72	3.44	3.13	4.14
	Tight	96.22	3.40	4.71	3.82
 (padded)	Relaxed	96.69	9.01	3.38	5.80
	Tight	98.01	2.63	2.58	3.04
	Relaxed	95.41	6.63	5.79	6.30
	Tight	93.61	8.72	7.00	7.35
	Relaxed	96.66	6.27	4.27	5.85
	Tight	88.98	11.41	12.22	10.83
 (padded)	Relaxed	94.80	8.87	6.11	7.80
	Tight	89.28	8.41	11.72	6.46
 (padded)	Relaxed	97.06	7.17	3.72	7.24
	Tight	97.17	7.60	2.93	4.98
Layouts without redundancy					
	Relaxed	46.00	22.46	31.98	18.28
	Tight	31.37	14.40	60.29	12.28
	Relaxed	11.33	11.30	50.59	20.60
	Tight	14.85	10.05	67.78	10.49

Table 1: Results for layouts with different degrees of redundancy under two schedule distributions.

collapses without converging to acceptable performance levels. More restrictive optimization problems induce harder underlying MDPs which destabilizes the RL training [Hafner et al. 2023]. The less restrictive scenarios (rows 1-6) almost converge near 10k episodes while the most restrictive scenarios progress slower by an order of magnitude. We hypothesize that the performance degradation in the more restrictive scenarios has to do with the increasing planning complexity arising from the restrictive maneuverability of the layouts. Examples of failed instances are shown in figure 3. These restricted layouts yield configurations that require long-horizon swapping maneuvers with higher degrees of agent coordination, and we look to further explore…

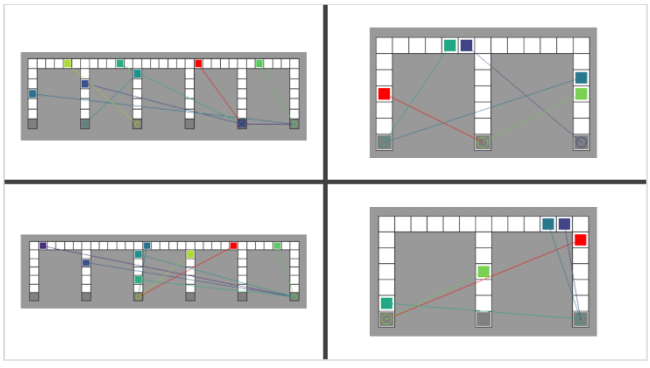

Figure 3: Example failure cases. Agents are represented by squares with lines pointing to their scheduled goal locations.

7 Conclusions

In this paper, we proposed a variant of the MAPF problem (MAPF-SC) to address scheduling constraints using shared rewards and spatiotemporal observations. We evaluated our algorithm on a variety of automation layouts and schedule distributions, and observed how the spatial constraints present a harder challenge than the temporal constraints. As next steps, we aim to improve agent coordination to improve performance in the more challenging layouts.

Disclaimer

The concepts and information presented in this paper are based on research results that are not commercially available. Future commercial availability cannot be guaranteed.

References

[Chudý et al. 2021] Chudý, J.; Surynek, P. 2021. ESO-MAPF: Bridging Discrete Planning and Continuous Execution in Multi-Agent Pathfinding. Proceedings of the AAAI Conference on Artificial Intelligence 35: 16014–16016.

[Damani et al. 2021] Damani, M.; Luo, Z.; Wenzel, E.; Sartoretti, G. 2021. PRIMAL$_2$: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning - Lifelong. IEEE Robotics and Automation Letters 6: 2666–2673.

[Gao et al. 2023] Gao, J.; Liu, Q.; Chen, S.; Yan, K.; Li, X.; Li, Y. 2023. Multi-agent path finding with time windows: Preliminary results. Proceedings of the 2023 international conference on autonomous agents and multiagent systems: 2586–2588.

[Hafner et al. 2023] Hafner, D.; Pasukonis, J.; Ba, J.; Lillicrap, T. 2023. Mastering diverse domains through world models. ArXiv preprint arXiv:2301.04104.

[Hoenig et al. 2016] Hoenig, W.; Kumar, T.K.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; et al. 2016. Multi-Agent Path Finding with Kinematic Constraints. Proceedings of the International Conference on Automated Planning and Scheduling 26: 477–485.

[van Knippenberg et al. 2021] van Knippenberg, M.; Holenderski, M.; Menkovski, V. 2021. Time-constrained multi-agent path finding in non-lattice graphs with deep reinforcement learning. Proceedings of the 13th Asian conference on machine learning 157: 1317–1332.

[Laurent et al. 2020] Laurent, F.; Schneider, M.; Scheller, C.; Watson, J.; Li, J.; Chen, Z.; et al. 2021. Flatland competition 2020: MAPF and MARL for efficient train coordination on a grid world. Proceedings of the NeurIPS 2020 competition and demonstration track 133: 275–301.

[Li et al. 2021] Li, J.; Tinka, A.; Kiesel, S.; Durham, J.W.; Kumar, T.K.S.; Koenig, S. 2021b. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. Proceedings of the

AAAI Conference on Artificial Intelligence 35: 11272–11281.

[Mohanty et al. 2020] Mohanty, S.; Nygren, E.; Laurent, F.; Schneider, M.; Scheller, C.; Bhattacharya, N.; et al. 2020. Flatland-RL : Multi-Agent Reinforcement Learning on Trains.

[Perron and Furnon 2023] Perron, L.; Furnon, V. 2023. OR-Tools.

[Sartoretti et al. 2019]Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.K.S.; Koenig, S.; et al. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. IEEE Robotics and Automation Letters 4: 2378–2385.

[Shahar et al. 2021] Shahar, T.; Shekhar, S.; Atzmon, D.; Saffidine, A.; Juba, B.; Stern, R. 2021. Safe Multi-Agent Pathfinding with Time Uncertainty. Journal of Artificial Intelligence Research 70.

[Surynek et al. 2019] Surynek, P.; Kumar, T.K.S.; Koenig, S. 2019. Multi-agent Path Finding with Capacity Constraints. In: Alviano, M.; Greco, G.; Scarcello, F. (Eds.), AI*IA 2019 – Advances in Artificial Intelligence, Vol. 11946, Springer International Publishing, Cham, p.235–249.

[Wang et al. 2022] Wang, H.; Chen, W. 2022. Multi-Robot Path Planning With Due Times. IEEE Robotics and Automation Letters 7: 4829–4836.

[Yu et al. 2022] Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; et al. 2022. The surprising effectiveness of PPO in cooperative multi-agent games. Advances in neural information processing systems 35: 24611–24624.

[Zhang et al. 2022] Zhang, H.; Chen, J.; Li, J.; Williams, B.C.; Koenig, S. 2022. Multi-agent path finding for precedence-constrained goal sequences. Proceedings of the 21st international conference on autonomous agents and multiagent systems: 1464–1472.