Multi-Agent Path Finding for Schedule Constrained Automation

Kareem Eissa, Rayal Prasad and Ankur Kapoor

***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. However, these usually require manual engineering efforts to use in practical applications that involve scheduling constraints. 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 rewards. Our approach is easily extendable to different constraints with simple adjustments to the reward design. We investigate the effects of temporal and topological variations of various automation scenarios on the performance of our method.**

I. 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 as shown in Fig. 1. Whether it is logistics, manufacturing, or even laboratory automation, the Multi-Agent Path Finding (MAPF) formulation can be used to address important as-pects of these target domains such as collision avoidance, travel time minimization, to ensure optimal and safe motion between machines. Recent works have shown significant progress in different variants of MAPF such as the lifelong version with a continuous stream of goals [7], the robust version with uncertainty in the envi-ronment execution [27] and executing the abstracted solution onto real systems [3].

One key assumption that is prevalent across MAPF variants is that task processing is instantaneous, which is atypical in most real-world situations. Even in the popular warehouse setting, a robot agent delivering a payload would require a non-instantaneous amount of time at the processing station. This discrepancy is amplified in domains such as factory automation, where tasks demand 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 and power requirements. Machines may also need additional time for inter-task requirements such as component switching or decontamination, usually enforced by the automation pipeline capabilities. Finally, a job would have a specific sequence of operations to undergo while machines may expect jobs to arrive in a particular order, leading to strong precedence constraints.

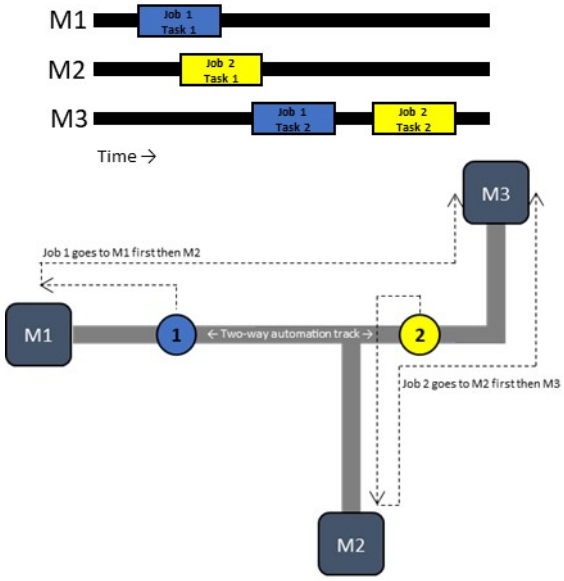


Fig. 1. Example automation scenario. **Top:** Automation jobs consisting of multiple tasks with multiple agents. **Bottom:** Agents (circles) need to plan collision-free routes to their goals (rectangles) according to the schedule (Note that machine M3 would expect Job 1 to arrive before Job 2).

Existing approaches to solving MAPF with such temporal aspects have shown promising results (including with a search-based CBS back bone [26]) but mostly involve designing heuristics and complex data structures to accommodate specific temporal and precedence constraints. Consequently, it is not straightforward to extend these approaches to the constraints and requirements of other domains. Furthermore, most are developed using simple static layouts usually modeled on video games or uniformly distributed warehouses.

In this work, we formulate MAPF-SC (MAPF with Scheduling Constraints) as a sequential decision-making RL problem that builds upon recent works such as [20], [21], [6], and [25]. Our approach incorporates spatio-temporal encodings and reward mechanisms that can be used to accommodate constraints specific to each use-case. Moreover, MAPF simulators that support temopral aspects are typically only designed for specific domains such as [5]. To address this, we also design a simulator that can be generalized to arbitrary domains with varying temporal constraints.

Kareem Eissa, Rayal Prasad and Ankur Kapoor are with Siemens Healthineers Digital Technology and Innovation, 755 College Road East, Princeton NJ 08540, USA {kareem.abdelrahman,rayal.prasad,ankur.kapoor}@siemens-healthineers.com

Contributions: In this work, we utilize multi-agent reinforcement learning (MARL) to solve the MAPF with scheduling constraints problem. Our contributions are summarized as follows:

- We model the MAPF-SC problem in the MARL framework and design spatio-temporal encodings of the scheduling constraints as inputs to the neural network.
- We develop a configurable procedural layout generator that captures real-world restrictions and guidelines.

To investigate the performance of our MARL method, we study the effects of spatial and temporal characteristics using a set of domain configurations inspired by real-world automation configurations.

II. 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 in an optimal [1], bounded suboptimal [2], and scalable [8] manner. There is also work focusing on adapting solution methods to incorporate constraints into the classical setup [13], [15]. The related literature we focus on involves MAPF variants that factor in time, as well as Reinforcement Learning approaches to the problem

A. MAPF with Time

Several works have studied temporal aspects of MAPF. In their recent work, Wang et al [17] focus on minimizing the violation of task specific due times and formulate three different objectives: Maximum Lateness to capture the worst violation of due times, Total Tardiness to capture the total violation of due times and Total Unit Penalties to capture the number of tasks with violated due times. More recently, Gao et al. [16] focus on the maximizing the average customer satisfaction proportional to the degree of deadline violation. They model time-windows to represent a transition in violations starting with a constant satisfaction of 1 until the early arrival time, after which the satisfaction is a decaying function proportional to the lateness. In other related work, Ma et al. [14] focus on the problem of maximizing the number of agents that reach their goals within a global deadline, while Zhang et al. [12] focus on the problem of satisfying precedence-constraints among goal sequences. While most of the previously cited literature concentrate on one-shot MAPF, Li et al. [7] propose a framework for solving lifelong MAPF by modeling it as a sequence of windowed MAPF instances.

The Flatland environment [5] simulates the railway setting as a multi-agent problem with the additional constraints of non-backward motion, arrival deadlines, and departure schedules. However, trains are allowed to park off the map before their departure and after their arrival, providing a degree of variation in the number of agents in the map. Our approach represents a more general automation scenario wherein agents maintain a continuous presence on the track, keeping the agent count constant.

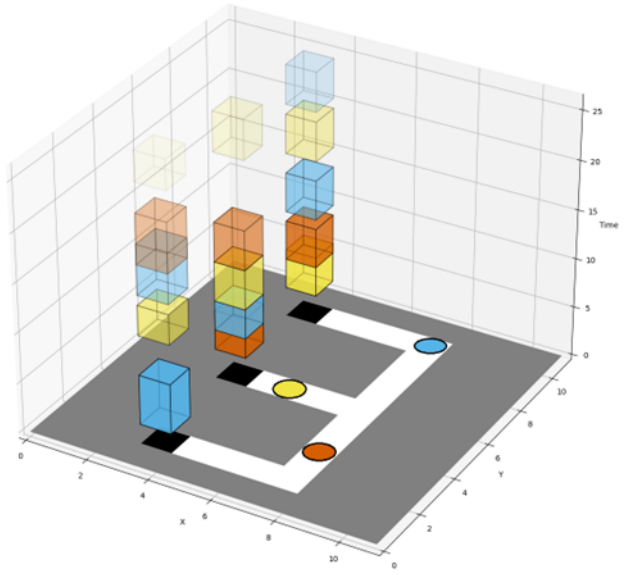


Fig. 2. Example MAPF-SC instance. The bottom XY plane encodes the spatial information. Agents are represented by circles, goal locations are represented by black squares, and gray regions are not traversable. Rectangles along the Time dimension represent scheduled goals for the respectively colored agents. The position of the rectangles represents the earliest arrival time, and its height represents the duration of the time-window. Processing times are implicit within the time windows for illustrative purposes.

More recently, CBS-TA-PTC [26] solves MAPF with some specific precedence and temporal constraints using a combination of data structures, heuristics, and an optimization solver. However, these constraints are particular for the bomb defusal task and the approach is not easily extendable to other domains. Furthermore, the one-shot nature of the algorithm provides practical challenges since it is common for automation scenarios to assign tasks sequentially. Our proposed framework addresses both of these – a life-long approach that can be easily extended to the unique temporal constraints of any domain.

B. Reinforcement Learning for MAPF

A number of recent works focus on solving MAPF with Reinforcement Learning (RL). Knippenberg et al. [25] aim to solve MAPF as a single agent RL problem that incorporates time constraints on the start and end of tasks. Similar to the Flatland environment, they provide a dummy starting node for each agent to wait until its starting time to be deployed onto the environment. Li et al. [6] propose learning decentralized MAPF policies aggregated with a graph neural network using imitation learning from expert search-based solvers. Sartoretti et al. [20] model MAPF as an MARL problem and utilize expert training to train decentralized policies with a mixture of imitation learning and RL. In an extension to this, Damani et al. [21] apply the decentralized policy paradigm to the lifelong variant of MAPF. More recently, Wang et al. [23] studied the effects of agent communication and the size of the local field of view on the decentralized MARL approach for one-shot MAPF. Our work builds on these

approaches and uses spatio-temporal encodings to allow real-world scheduling constraints to be incorporated.

III. Problem Formulation

The MAPF-SC problem is defined by an undirected graph $G = (V, E)$, a set of m agents $i_1, ..., i_m$, and a schedule S. An example instance is illustrated in Fig. 2. 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. Each agent i is assigned a sequence of goals corresponding to vertices $v_i^1, v_i^2...$ according to the schedule. We identify some key temporal elements that can be used to compose scheduling constraints for different automation domains. Each element can be captured by a soft or hard constraint to help model different scenarios. Specifically, we incorporate:

- Earliest Arrival Time $A_t(i, j)$ where an agent i is not allowed at the goal location j before this time.
- Deadline $D_t(i, j)$ where an agent i must be at the goal location j no later than this time.
- Goal Processing Time $P_t(i, j)$ where the agent i is stationary at the goal location j for this duration.
- Precedence constraints where multiple agents $i_1, ..., i_m$ assigned to the same goal location j have to be processed in a specific order.

Fig. 3 demonstrates the relationship between these temporal components. Different scenarios can be represented by adjusting these constraints - for example, classical MAPF has $A_t(i, j) = D_t(i, j)$ and $P_t(i, j) = 0$. Another popular domain is train scheduling [4], [5] where trains have a departure time modeled as $P_t(i, j)$ and $A_t(i, j) = 0$ for arrival as early as possible but leave the station by $D_t(i, j)$. There are no explicit precedence constraints in this scenario, only those captured by the train timetable i.e., in case of delays, precedence among trains can be violated. In conjunction with modifying the rewards, this gives us a simple yet powerful way to adapt the learned behavior of the agents to any domain's requirements. This ability to further modify constraints through the reward function is a key advantage of using an RL framework as it requires less manual efforts than custom heuristics and data structure design.

Fig. 3. Different temporal elements of scheduling constraints.

IV. 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 restrict our environments to 2D grids where each agent is limited to a local field of view observation [11], [19], [20]. We aim to learn a homogenous policy that can be deployed to different numbers of agents and executed in a decentralized manner.

A. Observations

The observation space consists of two main components to capture spatial and temporal information. The first component is the agent's Local Field of View (FOV) which builds upon earlier related works [20], [21]. This FOV is centered on the agent's current position and consists of channels denoting obstacles, the presence other agents, other agents' target locations, the direction towards the agent's goal, and a distance heatmap. Finally, a queue channel encodes the number of other agents that share the same target.

The second component directly encodes the agent's upcoming task. The spatial context of the task is represented through a vector, G_s, as the difference in x/y-coordinates and the goal distance. We clip this value to a maximum absolute value of 60 similar to Damani et al. [21]. Concurrently, the temporal aspects are captured in another vector, τ_t, that provides the remaining time until the earliest arrival time A_t, the duration until the deadline D_t, and a binary indicator if the deadline D_t has passed, marking the agent's delay.

TABLE I
Reward design under different scenarios

Event	MAPF-SC	Flatland	MAPF
At goal in $[A_t, D_t]$	+1.0	+1.0	+1.0
At goal before A_t	-0.1	0.0	0.0
Not at goal after D_t	-0.1	-0.1	-0.1
Shared any goal reward	+0.2	+0.2	+0.2
Shared any delay reward	-0.04	-0.04	0.0

B. 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 scheduled 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 the travel distance may be detrimental to the throughput due to induced congestion.

In Table I, we show reward realizations for different constraints in classical MAPF and Flatland [5]. Note how, for instance, constraints on early arrivals are relaxed in the MAPF and Flatland columns by simply removing the negative reward. In our experiments, we focus on the full problem of MAPF-SC without any relaxations (as shown in the first column).

In our setup, each agent receives a positive reward when it reaches its goal within the scheduled time-window. In contrast, it receives a negative reward for each time step it occupies its goal before its earliest arrival time A_t, to deincentivize congestion at the goal vertices. Similarly,

agents receive a negative reward for each time step it has not reached its goal past the deadline D_t to help incentivize arriving on time.

C. Training Setup

We process the local FOV with a three-layer convolutional neural network (CNN), each layer followed by a max-pooling operation (except for the last layer which is followed by a global average pooling operation) to produce a FOV representation vector. The task vectors (location and time) are subsequently fed through fully connected layers to produce a combined task representation vector. Both the FOV and task representation 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 (PPO) [22] which has shown great results in cooperative multi-agent settings [24].

V. Experiments

We design specific scenarios to examine how distinct spatial and temporal elements impact model performance. Additionally, we provide a comprehensive overview of the layout design's attributes and its influence on input spatial characteristics. Following this, we adjust the time-windows' parameters to induce variations in the resulting schedule slack.

A. Layouts

Most automation plants are restricted by empty space, machine dimensions, and other safety constraints and human factors. To investigate the role of different aspects on performance, we experiment with a set of carefully designed layouts as shown in the Layout column in Table V-B. These were inspired by real-world domains (flexible manufacturing and laboratory diagnostics) where 3-6 machines are usually placed in straight assembly-line like layouts. The designed layouts capture variations in restricted tracks connected by single lane corridors. 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 and large layouts with three and six machines/lanes respectively. 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.

B. Schedules

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 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 [4], [5]:

- Single-agent shortest distance (A* factor) which is the minimum distance traveled by an agent assuming there are no other agents in the goal.
- Multi-agent estimated congestion 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. Fig. 4 illustrates the 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 [18] where the solver optimizes for a randomly sampled list of agent tasks.

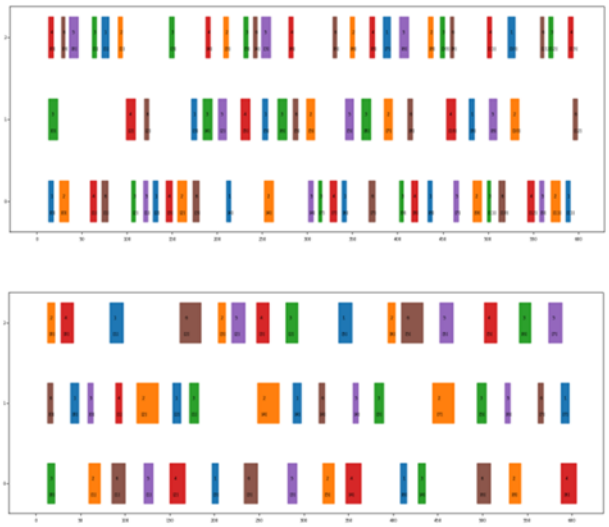


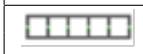
Fig. 4. Gantt charts of example schedules. Time is represented on the X-axis and goal locations (three in this case) are represented on the Y-axis. Tasks are represented as rectangles with their x-coordinate indicating their earliest start time and their width indicating duration. Colors correspond to agents executing each task. **Top**: Tight parameterization. **Bottom**: Relaxed parameterization.

C. Metrics

The primary objective of MAPF-SC is to maximize the number of goals completed by an agent within its scheduled time-windows. We measure the relative *throughput* of goals completed within the scheduled time-windows normalized by the ideal schedule execution. This adjusts the comparison

TABLE II

RESULTS FOR CLOSED LOOP AND DEAD END LANE LAYOUTS WITH OPTIONAL PADDING UNDER TWO SCHEDULE DISTRIBUTIONS

Layout	Size	Padding	Redundancy	Schedule	Total Finishes %		On-Time Finishes %		Late Finishes %	
					Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
	6	✗	✗	Relaxed	76.26	26.85	46.00	22.46	31.98	18.28
	6	✗	✗	Tight	90.90	10.01	31.37	14.40	60.29	12.28
	3	✗	✗	Relaxed	61.85	23.00	11.33	11.30	50.59	20.60
	3	✗	✗	Tight	82.53	10.23	14.85	10.05	67.78	10.49
	6	✓	✗	Relaxed	99.53	4.60	94.80	8.87	6.11	7.80
	6	✓	✗	Tight	99.42	6.24	89.28	8.41	11.72	6.46
	3	✓	✗	Relaxed	99.90	2.89	97.06	7.17	3.72	7.24
	3	✓	✗	Tight	99.52	6.04	97.17	7.60	2.93	4.98
	6	✗	✓	Relaxed	99.74	3.91	95.41	6.63	5.79	6.30
	6	✗	✓	Tight	99.59	5.08	93.61	8.72	7.00	7.35
	3	✗	✓	Relaxed	99.71	4.11	96.66	6.27	4.27	5.85
	3	✗	✓	Tight	99.57	3.95	88.98	11.41	12.22	10.83
	6	✓	✓	Relaxed	99.99	0.00	97.72	3.44	3.13	4.14
	6	✓	✓	Tight	99.99	0.06	96.22	3.40	4.71	3.82
	3	✓	✓	Relaxed	99.19	6.72	96.69	9.01	3.38	5.80
	3	✓	✓	Tight	99.99	0.11	98.01	2.63	2.58	3.04

to help understand the effects of different time-window distributions on the performance of the RL models. We report the mean and standard deviation across 1000 randomly generated schedules.

VI. 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.

In the closed-loop layouts (layouts 5-8 in Table V-B), 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 (layouts 3 and 4 in Table V-B), 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 the most restrictive scenarios i.e., unpadded layouts with dead end lanes (rows 1 and 2 in Table V-B), we observe a noticeable drop in performance compared to the other scenarios.

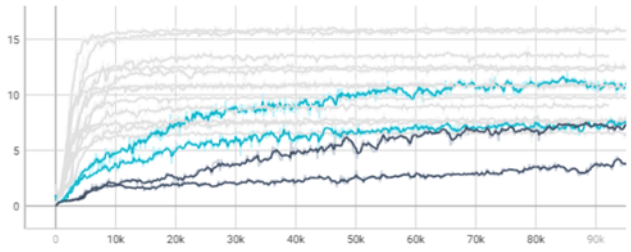


Fig. 5. Average completed tasks during training. Light and dark blue colors correspond to Layout 1 and 2 in Table V-B respectively. Light gray corresponds to the remaining layouts in Table V-B Tight schedules have a higher absolute number of tasks than their relaxed counterparts for the same finite horizon.

However, the most restrictive scenarios (layouts 1 and 2 in Table V-B) suffer from performance degradation. Counter-intuitively, their performance is better on the tight schedules than on the relaxed ones. In general, we find these instances to be less stable in training and the RL policy often collapses without converging to acceptable performance levels. More restrictive optimization problems induce harder underlying MDPs which destabilizes the RL training [9].

The training curves in Fig. 5 contrast the difference in convergence between the most restrictive scenarios and all other scenarios. The less restrictive scenarios (layouts 3-8 in Table V-B) 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. These restricted layouts yield configurations that require long-horizon swapping maneuvers with higher degrees of agent coordination.

VII. PROCEDURAL LAYOUT GENERATION

To extend our methods to more realistic automation domains, we incorporate the layout configuration parameters from our experiments and design a procedural layout generation algorithm. The generator also considers other uncontrollable aspects present in most automation plants such as non-empty spaces, operational restrictions, and safety constraints. For example, a machine may require special power consumption or ventilation which limits on its placement as well as input and output "lanes" connected to it. Our generator is parameterized by the following:

- Layout: Determines the dimensions of the floorplan. Optionally random sample permanent obstacles such as

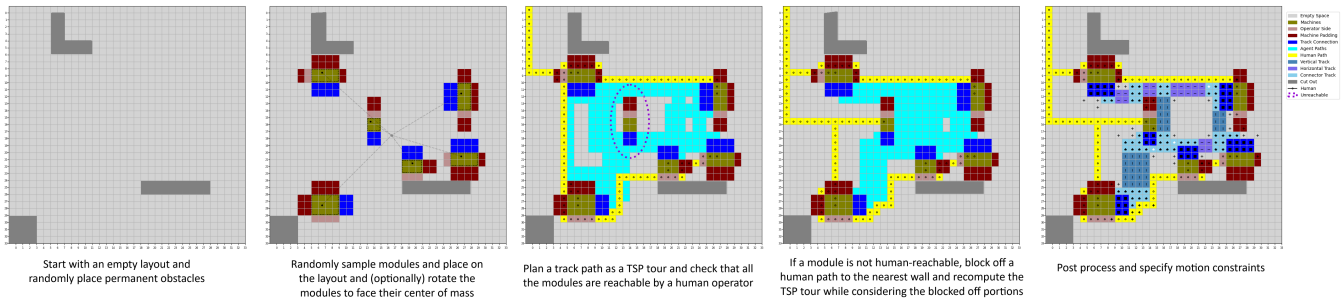


Fig. 6. Layout generation procedure example

building columns where no machines nor humans may traverse.

- Machine: Randomly sample locations with dimensions and accessibility requirements that affect its orientation as well as inputs and outputs.
- Lanes: Determine redundancy and padding as well as degrees of freedom in motion e.g., single direction conveyor belt. Optionally post processed to trade off between initial cost and operational efficiency through redundancy.
- Schedules: Parameterization based on number of agents and shortest path factors as described in the experiments section.

The generation algorithm is as follows: We start by initializing a layout as an empty 2D grid and randomly sample some permanent obstacles such as building columns. These obstacles are masked out from the grid and are not traversable by any entity. Next, we initialize a set of machines as goal locations in the MAPF-SC instances, sampled from a distribution of sizes and orientations. An optional step is to orient the machines towards their center of mass such that the generated track lanes are circular.

To generate the track lanes we compute the pairwise shortest path between all modules using the track redundancy and padding parameters. However, this leads to an overly redundant track so we refine it by computing a tour formulated as the Traveling Salesperson Problem (TSP) problem. This ensures that all modules are inter-reachable with optimal usage of track lanes.

It is usually required (for maintenance purposes) that all machines are human accessible. However, in most cases, a human cannot walk the same lanes as the agents. This constraint may be relaxed in some domains such as warehouses where both humans and machines may walk on the same floors. During generation, some machines may be completely surrounded by track lanes and inaccessible for humans. To solve this issue we introduce additional obstacles for human traversal and re-plan the track lanes. Finally, this layout is used to generate schedules of different parameterizations as described previously in the experiments section. Fig. 6 illustrates the procedural generation process.

Our experiments and analysis on the restricted layouts in Table V-B highlight the importance of redundancy or ”padding” to allow for smooth agent flow. When we trained and evaluated on a variety layouts with redundancy, the models produced were capable of meeting almost all of the goals within their time-windows. We attribute this level of performance primarily to the layout as we observe that agents learn traffic-like behaviors such as using parallel lanes for opposite traffic flow.

By designing custom layouts and incorporating the flexible constraint composition of our proposed MAPF-SC approach, we provide a simple framework to adapt MAPF for any automation domain. One can easily shape the reward function to help agents learn the required behavior, while using the generator to create high-fidelity spatial and temporal environments to train these agents on.

VIII. Conclusion

In this paper, we proposed a variant of the MAPF problem to address scheduling constraints. We proposed a MARL framework to solve MAPF-SC using shared rewards and spatio-temporal observations. We studied aspects affecting the efficiency of our method under different layouts parameterizations and schedule distributions. Our empirical investigations showed that our model achieves near perfect schedule execution for most scenarios and identified hard constraints that hinder throughput. To generalize our findings to real-world automation domains, we design a configurable procedural layout generator that allows training custom RL models for different domain requirements.

IX. 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

[1] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, “Conflict-Based Search For Optimal Multi-Agent Path Finding,” AAAI, vol. 26, no. 1, pp. 563–569, Sep. 2021, doi: 10.1609/aaai.v26i1.8140.

[2] J. Li, W. Ruml, and S. Koenig, “EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding,” AAAI, vol. 35, no. 14, pp. 12353–12362, May 2021, doi: 10.1609/aaai.v35i14.17466.

[3] J. Chudý and P. Surynek, “ESO-MAPF: Bridging Discrete Planning and Continuous Execution in Multi-Agent Pathfinding,” AAAI, vol. 35, no. 18, pp. 16014–16016, May 2021, doi: 10.1609/aaai.v35i18.17997.

[4] F. Laurent et al., "Flatland Competition 2020: MAPF and MARL for Efficient Train Coordination on a Grid World," in Proceedings of the NeurIPS 2020 Competition and Demonstration Track, H. J. Escalante and K. Hofmann, Eds., in Proceedings of Machine Learning Research, vol. 133. PMLR, Dec. 2021, pp. 275–301. [Online]. Available: https://proceedings.mlr.press/v133/laurent21a.html

[5] S. Mohanty et al., "Flatland-RL: Multi-Agent Reinforcement Learning on Trains," 2020, arXiv. doi: 10.48550/ARXIV.2012.05893.

[6] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph Neural Networks for Decentralized Multi-Robot Path Planning," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 11785–11792. doi: 10.1109/IROS45743.2020.9341668.

[7] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses," in Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, in AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1898–1900.

[8] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search," AAAI, vol. 36, no. 9, pp. 10256–10265, Jun. 2022, doi: 10.1609/aaai.v36i9.21266.

[9] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," 2023, arXiv. doi: 10.48550/ARXIV.2301.04104.

[10] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-Aware Graph Attention Networks for Large-Scale Multi-Robot Path Planning," IEEE Robot. Autom. Lett., vol. 6, no. 3, pp. 5533–5540, Jul. 2021, doi: 10.1109/LRA.2021.3077863.

[11] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, et al., "Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks," 2023, arXiv. doi: 10.48550/ARXIV.2306.13831.

[12] H. Zhang, J. Chen, J. Li, B. C. Williams, and S. Koenig, "Multi-Agent Path Finding for Precedence-Constrained Goal Sequences," in Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, in AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2022, pp. 1464–1472.

[13] P. Surynek, T. K. S. Kumar, and S. Koenig, "Multi-agent Path Finding with Capacity Constraints," in AI*IA 2019 – Advances in Artificial Intelligence, vol. 11946, M. Alviano, G. Greco, and F. Scarcello, Eds., in Lecture Notes in Computer Science, vol. 11946. , Cham: Springer International Publishing, 2019, pp. 235–249.

[14] H. Ma, G. Wagner, A. Felner, J. Li, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with deadlines," in Proceedings of the 27th International Joint Conference on Artificial Intelligence, in IJCAI'18. Stockholm, Sweden: AAAI Press, 2018, pp. 417–423.

[15] W. Hoenig et al., "Multi-Agent Path Finding with Kinematic Constraints," ICAPS, vol. 26, pp. 477–485, Mar. 2016, doi: 10.1609/icaps.v26i1.13796.

[16] J. Gao, Q. Liu, S. Chen, K. Yan, X. Li, and Y. Li, "Multi-Agent Path Finding with Time Windows: Preliminary Results," in Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, in AAMAS '23. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2023, pp. 2586–2588.

[17] H. Wang and W. Chen, "Multi-Robot Path Planning With Due Times," IEEE Robot. Autom. Lett., vol. 7, no. 2, pp. 4829–4836, Apr. 2022, doi: 10.1109/LRA.2022.3152701.

[18] L. Perron and V. Furnon, OR-Tools. (May 07, 2024). Google. [Online]. Available: https://developers.google.com/optimization/

[19] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "POGEMA: Partially Observable Grid Environment for Multiple Agents," 2022, arXiv. doi: 10.48550/ARXIV.2206.10944.

[20] G. Sartoretti et al., "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning," IEEE Robot. Autom. Lett., vol. 4, no. 3, pp. 2378–2385, Jul. 2019, doi: 10.1109/LRA.2019.2903261.

[21] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL$_2$: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning - Lifelong," IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 2666–2673, Apr. 2021, doi: 10.1109/LRA.2021.3062803.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, arXiv. doi: 10.48550/ARXIV.1707.06347.

[23] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding," in Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, in AAMAS '23. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2023, pp. 2598–2600.

[24] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in Proceedings of the 36th International Conference on Neural Information Processing Systems, in NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[25] M. van Knippenberg, M. Holenderski, and V. Menkovski, "Time-Constrained Multi-Agent Path Finding in Non-Lattice Graphs with Deep Reinforcement Learning," in Proceedings of The 13th Asian Conference on Machine Learning, V. N. Balasubramanian and I. Tsang, Eds., in Proceedings of Machine Learnfing Research, vol. 157. PMLR, Nov. 2021, pp. 1317–1332. [Online]. Available: https://proceedings.mlr.press/v157/knippenberg21a.html

[26] Yu Quan Chong, Jiaoyang Li, Katia Sycara, "Optimal Task Assignment and Path Planning using Conflict-Based Search with Precedence and Temporal Constraints," in AAMAS '24: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2024, pp. 2210-2212.

[27] 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.