MAPF and MAPD: Recent Developments and Future Directions

Francesco Amigoni[1], Davide Azzalini[1], Nicola Basilico[2] and Benedetta Flammini[1]

[1]*Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria*
[2]*Università degli Studi di Milano, Dipartimento di Informatica*

Abstract
Multi-Agent Path Finding (MAPF) is the problem of computing collision-free paths for a group of agents such that they can safely reach their target locations. In Multi-Agent Pickup and Delivery (MAPD), pickup and delivery locations are provided at runtime for an ideally ever-ending stream of tasks, making MAPD a combination between MAPF and online task assignment. Current models for MAPF and MAPD do not consider many of the practical issues encountered in real applications: in this paper, we present four new variants of the MAPF and MAPD problems we have recently devised which try to overcome different shortcomings of the original formulations. Promising directions of future work are also identified.

Keywords
Multi-Agent Path Finding, Multi-Agent Pickup and Delivery, Multirobot Systems

1. Introduction

Multi-Agent Path Finding (MAPF) and its lifelong version, Multi-Agent Pickup and Delivery (MAPD), have gained a lot of attention from both academia and industry in recent years. The key objective in both problems is to compute paths for a team of agents moving in a shared environment such that they can perform different tasks while avoiding collisions. MAPD differs from MAPF in the fact that new (previously unknown) tasks are continuously added at runtime and have to be dynamically assigned to the agents for their execution. Each task prescribes to reach a goal (delivery) location from an initial one. In MAPD, agents have the additional requirement of passing through a pickup location before reaching their goal. MAPF and MAPD are encountered in several application domains. Notable examples include automated warehouses [1] where robots continuously fulfill new orders, coordination of service robots [2] or fleets of autonomous cars, and automated control of non-player characters in video games [3]. Being MAPF and MAPD relatively young problems, several extensions are being investigated by the research community, especially those that would bridge the gap between theory and applicability to real use cases. In this paper, we present four new variants of the MAPF and MAPD problems recently proposed by the authors with the aim of overcoming different shortcomings

*9th Italian Workshop on Artificial Intelligence and Robotics, in conjunction with the 21st International Conference of the Italian Association for Artificial Intelligence (AI*IA 2021), November 28 - December 2, 2022, Udine, Italy.*
Authors equally contributed to this paper and are listed in alphabetical order.

francesco.amigoni@polimi.it (F. Amigoni); davide.azzalini@polimi.it (D. Azzalini); nicola.basilico@unimi.it (N. Basilico); benedetta.flammini@polimi.it (B. Flammini)

of the original formulations and covering a wider range of real applications. Additionally, possible directions of future work are identified.

2. MAPF and MAPD

In this section, we present the original versions of MAPF and MAPD problems in order to set the background for our extensions.

2.1. MAPF

The Multi-Agent Path Finding (MAPF) problem consists in finding start-goal paths for a team of agents such that they do not collide with each other [4]. In the classical MAPF formulation with a team of k agents, the input is composed of an undirected graph $G = (V, E)$, whose vertices in V represent the locations of the environment and edges in E their connections. A function $s : \{1, \ldots, k\} \rightarrow V$ maps each agent to its starting vertex, while a function $t : \{1, \ldots, k\} \rightarrow V$, provides the agent's goal vertex. It is assumed that time is discrete and that at each time step each agent can perform an action $a : V \rightarrow V$. We distinguish between two different types of actions: at time t an agent at vertex v can either *move* to an adjacent vertex v' (i.e., $(v, v') \in E$), so that it will be at vertex v' at time $t + 1$, or it can *wait* in its location, remaining at vertex v at time $t + 1$. A sequence of actions $\pi_i = (a_1, \ldots, a_n)$ is called a single-agent path for agent i iff executing these actions from the source vertex $s(i)$ leads agent i to reaching the target vertex $t(i)$. A feasible solution for the MAPF problem is a set of k single-agent paths (one for each agent), such that the paths do not collide with one another when executed. Two types of collisions can happen: *vertex conflicts* happen when two different agents are at the same vertex at the same time step, while *swapping conflicts* occur when two different agents traverse the same edge in opposite directions at the same time. Given a solution $\pi = \{\pi_1, \ldots, \pi_k\}$, different objective functions can be used to evaluate its quality. The most used are the *makespan*, which considers the time steps needed so that all the agents reach their target ($\max_{1 \leq i \leq k} |\pi_i|$), and the *flowtime*, that is the sum of time steps needed by each agent to reach its target ($\sum_{i=1}^{k} |\pi_i|$).

2.2. MAPD

In many real-world applications, such as automated warehouses, agents have not just one task to perform, but receive an online stream of tasks. For this reason, a lifelong version of MAPF has been proposed, called Multi-Agent Pickup and Delivery (MAPD) [5]. The MAPD problem consists in finding collision-free paths for a team of agents that have to complete several tasks in an online manner. For this reason, MAPD combines collision-free path planning (i.e., MAPF) and online task assignment.

A MAPD problem for k agents involves an undirected connected graph $G = (V, E)$, that represents the environment, and a task set $\mathcal{T}$ that contains the tasks that have not yet been executed. The task set is dynamic, since new tasks can be added at any time. Each task $\tau_j = (s_j, g_j) \in \mathcal{T}$ is composed of a pickup location $s_j \in V$ and a delivery location $g_j \in V$. In order to complete a task, an agent has to reach g_j passing through s_j. When an agent does not have any task assigned, it is called *free* and it can be assigned to a task in $\mathcal{T}$, otherwise if the agent has an assigned task is called *occupied*. An occupied agent becomes free when it reaches the delivery location of its current task after passing from the pickup location. The objective of a MAPD problem is to complete all the tasks in the shortest possible time: the quality of a solution is evaluated using either the *service time*, that is the average number of time steps required to complete a task since its appearance, or the *makespan*, that is the smallest number of time steps necessary to complete all the tasks (assumed to be finite in number).

3. Some Recent Results

In this section, we outline the extensions to MAPF and MAPD that we recently proposed.

3.1. MAPF-C: MAPF in Configurable Environments

In almost all the existing formulations of the MAPF problem, it is assumed that the environment is given and static. However, in many real-world applications, this assumption does not hold: for example, in warehouse logistics shelves can be moved and their positions can be re-arranged. In [6], we propose *MAPF in Configurable environments* (*MAPF-C*), in which the presence and arrangement of the obstacles in the environment can be controlled under some constraints. Given a MAPF problem, the MAPF-C variant considers all the possible legal configurations of the environment and returns both the optimal environment configuration and a solution for the problem. We devise two different approaches to solve the MAPF-C problem, both based on Conflict Based Search (CBS) [7], an optimal and complete MAPF solver. The first algorithm is called Parallel CBS (P-CBS), in which the idea is to run in parallel the CBS algorithm on all the possible environment configurations and to stop the search when a least-cost solution is found. The second algorithm is called Abstract CBS (A-CBS), an extension of CBS that considers not only constraints for the agents but also for the environment. Both algorithms are optimal and complete. Experiments are conducted on a 15×15 grid map, placing 1×1 obstacles and allowing the solvers to remove 30% of the present obstacles, and on a large 128×128 grid map with 4×4 obstacles, each with an integer removal cost picked randomly from $\{1, 2, 3, 4, 5\}$, and a removal budget of 10. The settings are tested for a varying number of agents and obstacles. Both results show that A-CBS performs better than P-CBS in terms of runtime, instances solved, and computing effort, especially when the number of possible environment configurations and agents increases.

3.2. MAPD-d: MAPD with Delays

Current algorithms do not consider many of the practical issues encountered when executing planned paths; in fact, real agents often do not follow the paths as expected and may be subject to delays and failures. Recently, the MAPF community has focused on robustness, generally understood as a property of solutions that can withstand real-world-induced relaxations of some idealistic assumptions made by the models [8, 9]. A typical example is represented by the assumption that paths are executed without errors. In reality, however, path execution is subject to delays and other issues that can hinder some properties (e.g., the absence of collisions) of a solution. For the MAPD problem, robustness has yet to be consistently studied. We study the robustness of MAPD to the occurrence of delays by defining a variant of the problem that we call *MAPD with delays* (*MAPD-d*) [10]. In this variant, agents, like in the original MAPD, are assigned to tasks, which may continuously appear at any time step, and paths to accomplish those tasks while avoiding collisions are computed. During path execution, delays can occur at arbitrary times, causing one or more agents to halt at some time steps, thus slowing down the execution of their planned paths. We devise a set of algorithms to compute robust solutions for MAPD-d based on TP (a decentralized MAPD solver) [5], to which we add recovery routines that replan in case collisions caused by delays are detected. The two algorithms we propose adopt the approach of robust planning, computing paths that limit the risk of collisions caused by potential delays, providing deterministic (*k*-TP) and probabilistic (*p*-TP) guarantees, respectively. We test our algorithms in two warehouse grid environments: a small one, 15×13, with 4 and 8 agents, and a large one, 25×17, with 12 and 24 agents. We create a sequence of 50 tasks whose arrival time is determined according to a Poisson distribution. We test 3 different arrival frequencies λ for the tasks: 0.5, 1, and 3. During each run, 10 delays per agent are randomly inserted. Experiments show that our methods deliver robust MAPD solutions, greatly reducing the number of replans needed to cope with delays at the cost of a small increase in solution cost and runtime. For example, for a large warehouse with $\lambda = 3$, *k*-TP decreases the number of replans of more than 75% wrt plain TP with an increase in the total cost of less than 2%.

3.3. MAPD-P: MAPD with Task Probability Distribution

MAPD's general assumption is that the new tasks that are continuously added to the system are not known in advance. As a result, existing MAPD algorithms assume complete ignorance about the position and the time at which future tasks will appear until they are actually added to the system. This assumption can however be too strict in many real-world scenarios, such as warehouses, in which historical records can be used to predict where and when it is more likely that new tasks will appear in the future. Knowledge about when and where future tasks are expected to appear could be used to make better choices during both task assignment and path planning, which would in turn lead to reduce the average time required to complete tasks and thus improve the overall performance of the system. To this end, we define a new variant of the MAPD problem called *MAPD with task Probability distribution* (*MAPD-P*), in which we assume to know the probability distribution of appearance of tasks at specific time instants in the future and at specific locations [11]. To solve a MAPD-P problem, we define a set of algorithms extending TP in which agents take advantage of the knowledge of the task probability distribution to decide how to behave when they are free or when they are assigned to a new task. Tests are performed on two 37×25 warehouse grids, that differ in the arrangement of shelves. We test 4 different task frequencies as input to the task generation distribution: 0.05, 0.1, 0.2, and 0.5 tasks per time step. For each task frequency, we run a test with 30 tasks and a set of 10 agents and 4 tests with 80 tasks and different numbers of agents: 10, 20, 30, and 40 agents. Experiments show that the proposed algorithms can effectively exploit this knowledge by reducing the average time required to complete tasks, especially in settings with low task frequency (around 42% for low-frequency settings and around 4% for high frequency ones).

3.4. MAPD-g: MAPD for a Guest Team

Another recent development is the formulation of *MAPD for a guest team* (*MAPD-g*), a new MAPD variant in which there are *two* different teams of agents in the same environment, each one with its own MAPD problem [12]. A team, called guest team, has to solve its MAPD tasks without communicating or interfering with the other team, called home team, that is already addressing another MAPD problem. Possible applications of this setting can be found in warehouse logistics, for example when some locations need cleaning: while a team (home team) is performing the usual pickup and delivery tasks, an external team (guest team) is required to clean or remove items from the ground without interfering with the ongoing tasks. Adopting a strategy in which guests plan their paths without considering home robots and replan every time there is a potential collision can slow down the completion of tasks for guests. To overcome this issue, we propose guest agents to build a model of the behavior of the home team, and then incorporate this information in their planning phase. In order to model the behavior of the home robots, we consider two possible scenarios: the first one assumes that guests have a period of time during which they can observe home agents and construct a model according to what they observe. In the second scenario, guests model the behavior of home robots while performing their MAPD tasks, incrementally updating knowledge according to what they observe. Given this information, the aim in guests' planning phase is to find a balance between keeping the path length short and trying to avoid home robots. The method is tested on a 48×46 grid warehouse with 15 home agents and 3 guests, and a 73×41 video game map with 15 home agents and 5 guests. The length of guests' observation period is 300 time steps. Experiments carried on using a probabilistic occupancy model for the home team behavior show that the inclusion of this information in guests' planning phase reduces the number of collisions (29% for warehouse and 42% for video game) and decreases tasks' completion time (7% for warehouse and 5% for the video game) for guests.

4. Future Work

Classical algorithms that are used to solve MAPF problems usually have a common limitation: scalability [13]. Optimal algorithms usually start failing when the number of agents increases beyond the hundreds. Sub-optimal algorithms tend to scale better, but it is still hard to achieve good performance in terms of makespan or flowtime. The issue becomes even more evident when dealing with MAPD problems. For these reasons, recent years have seen a shift towards Machine Learning approaches. Taking inspiration from [14], we are investigating the application of Graph Attention Neural Networks in order to enable agents to learn different importance weights for messages coming from different neighbours, selecting who to listen to or not when deciding where to move. In this way, information about goals and future movements is shared amongst agents, improving coordination in local decision-making and avoiding collisions.

Acknowledgments

Authors are thankful to M. Bellusci, G. Lodigiani, A. Di Pietro, and A. Bignoli.

References

[1] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (2008) 9–20.

[2] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, Cobots: Robust symbiotic autonomous mobile service robots, in: Proc. IJCAI, 2015, pp. 4423–4429.

[3] H. Ma, J. Yang, L. Cohen, T. S. Kumar, S. Koenig, Feasibility study: Moving non-homogeneous teams in congested video game environments, in: Proc. AIIDE, 2017, pp. 270–272.

[4] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, et al., Multi-agent pathfinding: Definitions, variants, and benchmarks, in: Proc. SOCS, 2019, pp. 151–158.

[5] H. Ma, J. Li, T. Kumar, S. Koenig, Lifelong multi-agent path finding for online pickup and delivery tasks, in: Proc. AAMAS, 2017, pp. 837–845.

[6] M. Bellusci, N. Basilico, F. Amigoni, Multi-agent path finding in configurable environments, in: Proc. AAMAS, 2020, pp. 159–167.

[7] G. Sharon, R. Stern, A. Felner, N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artif. Intell. 219 (2015) 40–66.

[8] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, N.-F. Zhou, Robust multi-agent path finding, in: Proc. AAMAS, 2018, pp. 1862–1864.

[9] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, S. Koenig, Probabilistic robust multi-agent path finding, in: Proc. ICAPS, 2020, pp. 29–37.

[10] G. Lodigiani, N. Basilico, F. Amigoni, Robust multi-agent pickup and delivery with delays, arXiv preprint arXiv:2303.17422 (2023).

[11] A. Di Pietro, N. Basilico, F. Amigoni, Multi-agent pickup and delivery with task probability distribution, in: Proc. AAMAS, 2023.

[12] B. Flammini, D. Azzalini, F. Amigoni, Multi-agent pickup and delivery in presence of another team of robots, in: Proc. AAMAS, 2023.

[13] J. Yu, S. M. LaValle, Structure and intractability of optimal multi-robot path planning on graphs, in: Proc. AAAI, 2013, pp. 1443–1449.

[14] Q. Li, W. Lin, Z. Liu, A. Prorok, Message-aware graph attention networks for large-scale multi-robot path planning, IEEE RA-L 6 (2021) 5533–5540.