# State of the Art on: Multi-Agent Pickup and Delivery

Andrea Di Pietro, andrea4.dipietro@mail.polimi.it

## 1. Introduction to the research topic

Multi-Agent Pickup and Delivery (MAPD) is a problem in the field of multi-robot systems in which agents have to execute a continuously updating set of tasks, each one consisting in moving from a pickup location to a delivery location in a predefined environment avoiding collisions with other agents. MAPD is an extension of another problem, Multi-Agent Path Finding (MAPF), in which tasks are preassigned to agents (one task per agent) and can't be added during the execution.

**Conferences and Journals**

Considering parameters like the H5-index of publications, the Impact Score [1] and the average acceptance rate, the most relevant conferences with respect to our problem are:

- *Association for the Advancement of Artificial Intelligence (AAAI)*
- *International Joint Conference on Artificial Intelligence (IJCAI)*
- *International Conference on Automated Planning and Scheduling (ICAPS)*
- *International Conference on Robotics and Automation (ICRA)*
- *International Conference on Intelligent Robots and Systems (IROS)*
- *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*

and the most relevant journals with respect to our problem are:

- *Artificial Intelligence Journal (AIJ)*
- *Journal of Artificial Intelligence Research (JAIR)*
- *Autonomous Robots (AURO)*
- *IEEE Transactions on Robotics (T-RO)*
- *International Journal of Robotics Research (IJRR)*
- *Robotics and Autonomous Systems (RAS)*

### 1.1. Preliminaries

The MAPF and MAPD problems are defined on an *undirected graph*. An undirected graph G = (V, E) is a mathematical structure that consists of a set of nodes V and a set of pairs of nodes (or edges) $E \subseteq V^2$.

[1] The number of citations received in that year of articles published in a specific journal during the two preceding years, divided by the total number of publications in that journal during the two preceding years.

### 1.1.1 MAPF

A classical MAPF problem [13] is defined on an undirected graph *G = (V, E)* and a set of *k* agents. Each agent is assigned a source vertex $s_i$ and a target vertex $t_i$. Time is assumed to be discrete, and in every time step each agent occupies one of the graph vertices and can perform an action.

**Definition 1.** *An action is a function a: $V \rightarrow V$. Each agent can perform two different types of actions: moving and waiting. A moving action changes the current vertex v of the agent to an adjacent vertex v', i.e., a vertex v' so that $(v, v') \in E$. A waiting action is an action such that $a(v) = v$.*

**Definition 2.** *A single-agent path $\pi_i$ for an agent i is a sequence of actions that executed from $s_i$ leads to $t_i$.*

A solution for a MAPF problem is a set of k single-agent paths without collisions. There exist different kinds of collisions.

**Definition 3.** *A vertex collision between two paths $\pi_i$ and $\pi_j$ of two agents occurs when there exists a time step t such that the vertex occupied by agent i at time step t ($\pi_i[t]$) is equal to the vertex occupied by agent j at the same time step t ($\pi_j[t]$).*

**Definition 4.** *A swap collision between two paths $\pi_i$ and $\pi_j$ of two agents occurs when the two agents swap locations in a single time step, so that there exists a time step t such that $\pi_i[t+1] = \pi_j[t]$ and $\pi_i[t] = \pi_j[t+1]$.*

MAPF solutions are evaluated through an objective function to be minimized such as the *makespan* or the *sum of costs*.

**Definition 5.** *The makespan is the number of steps after which all agents have reached their targets without collisions.*

**Definition 6.** *The sum of costs (or flowtime) is given by the sum of the numbers of steps of each single path planned by each agent.*

Finding an optimal solution for a MAPF problem is NP-hard for both the aforementioned optimization criteria [14][17].

### 1.1.2 MAPD

A MAPD problem [8] is equally defined on a undirected graph and a set of agents, where each agent is associated to an initial vertex. In addition a set $T$ is defined. $T$ is the set of unexecuted tasks and it is updated at each time step with the new incoming tasks. Each task $t_i \in T$ is associated to a pickup location $s_i \in V$ and a delivery location $g_j \in V$.

**Definition 7.** *An agent is free if and only if it is not executing any task.*

Free agents can be assigned to unexecuted tasks and they have to move to the delivery location of the task passing through its pickup location. A solution of a MAPD instance is a plan such that all tasks are executed in a bounded amount of time after they have been inserted into the system [7]. Solutions can be evaluated through objective functions to be minimized such as the *makespan* or the *service time*.

**Definition 8.** *The makespan is the number of steps after which all the tasks in T have been completed.*

**Definition 9.** *The service time is the average number of steps between the addition of a task to the system and the completion of the task itself.*

As an extension of MAPF, MAPD is necessarily NP-hard.

### 1.2. Research topic

The Multi-Agent Path Finding problem is an abstract representation of the common real problem of agents that need to move in an environment to reach a goal by avoiding collisions and maximizing the efficiency. This problem applies to many real applications [4] such as autonomous vehicles, video games, and warehouses. However, MAPF is a simplification of real word problems, where tasks are not always known and assigned in advance but agents need to be assigned and to execute in sequence new incoming tasks whose pickup and delivery locations and arrival time can be uncertain or completely unknown both in the distribution across the environment and in the frequency. Multi-Agent Pickup and Delivery is a generalisation of MAPF that better fits real life scenarios such as automated warehouses where mobile robots move items from a location to another.

## 2. Main related works

### 2.1. Classification of the main related works

The MAPD problem has been addressed in different works by using different assumptions or approaches. Here some of the main aspects that can differentiate the proposals that have been formulated from works about MAPD problems and solutions are presented.

**Planning methods** Different methods to decompose the problem and to define the planning strategy have been proposed [4]:

- Solving the MAPD problem by knowing all the tasks in advance and working in an offline setting where the solution is computed once and does not change during the execution of the MAPD instance [5][9].
- Decomposing the MAPD problem into a sequence of MAPF instances where all paths are replanned at every time step to take into account the new tasks [16].
- Decomposing the MAPD problem into a sequence of MAPF instances where the path replanning is performed only for some of the agents at each time step, for example only for the agents that have reached their current goal location or in the most extreme scenario for one agent at the time [8].

**Centralized and decoupled algorithms** In centralized algorithms the task assignment and the path planning are centralized, while in decoupled algorithms each agent assigns itself to tasks and computes its own path [8].

**Completeness and optimality of solutions** Some algorithms can guarantee the completeness or the optimality of the solution while some others do not. Some algorithms guarantee a sub-optimal upper bound to the objective function that they aim to minimize.

**Tasks probability distribution** Experimental results can be influenced by the way in which the tasks are introduced in the environment and some works can aim to optimize the performance of an algorithm in a specific configuration of the environment or for a specific kind of probability distribution of the pickup and delivery locations of tasks across the environment.

The discussion about the aforementioned approaches has not led to the identification of a best choice among the different ways to decompose the problem and the different types of algorithms because every choice is a compromise between the optimality and completeness of some solutions and the better scalability of some other sub-optimal solutions in scenarios with a large number of agents and tasks.

## 2.2. Brief description of the main related works

### 2.2.1 Algorithms

All the algorithms for MAPD problems take advantage in some way of the contribution of existing MAPF algorithms to perform the agents path planning after the tasks have been assigned (the same is not possible for tasks assignment because in MAPF problems tasks are preassigned to agents). Some of the most commonly used algorithms are:

- *Conflict Based Search (CBS)* [11]: A two-level complete and optimal MAPF algorithm. At the high level, a search is performed on a Conflict Tree. Each node in the tree represents a set of constraints on the possible movements of the agents. At the low level, each agent tries to find a path that satisfies the constraints imposed by the high-level node that is currently being examined. The nodes are analyzed in increasing order of cost of the associated plan in order to grant the optimality of the solution.
- *Increasing Cost Tree Search (ICTS)* [12]: A two-level complete and optimal algorithm. The high-level phase of ICTS iterates on a cost tree starting from the minimum possible cost and and increasing the expected cost of the path of each single agent. The low-level phase tries to find a collision-free path for every agent with the given cost.
- *Prioritized Planning* [6]: A neither optimal nor complete algorithm in which agents do not plan their paths simultaneously but in a specific order. The first agent can choose the optimal path regardless of other agents and each subsequent agent in the set will choose its optimal path by considering the paths planned by the previous agents in the sequence as constraints.

The properties of the different algorithms that have been proposed are influenced by the aforementioned choice concerning how to decompose the problem, and in particular the difference between offline and online MAPD formulations. While in offline MAPD the arrival time steps, the pickup and the delivery locations of tasks are known a priori, in online MAPD the new tasks appear to the system only at their arrival time step and they can't be considered by the algorithm in advance.

For offline MAPD two algorithms have been proposed that use respectively Prioritized Planning and CBS: *Task Assignment and Prioritized path planning (TA-Prioritized)* and *Task Assignment and Hybrid path planning (TA-Hybrid)* [5]. Both algorithms assign the tasks by solving a special Travel Salesman Problem, which ignores collisions and uses estimated travel times between locations in order to minimize the *makespan*, then the algorithms plan collisions-free paths for the agents that visit pickup and delivery locations in the order of the tasks. The two algorithms differ in their path planning methods and thus in their trade-off between efficiency and effectiveness. These algorithms can be applied only when the pickup and delivery locations and the arrival times of all tasks are known in advance.

For what concerns online algorithms, Ma et al. [8] have proposed three different algorithms, two of which are decoupled while the third one is centralized.

- *Token Passing (TP)* is a decoupled algorithm based on a token, a data structure that is shared among the agents and that contains the current paths of all agents and the task set $T$ with the current assignment status. System initializes the token with the trivial paths where all agents wait at their initial vertices. At each time step, the system adds all new tasks to the task set. Each free agent requests the token once per time step and they choose an unassigned task from the task set such that no path of other agents in the token ends at the pickup or delivery vertex of the task. The task with minimum path cost is chosen. Finally, the agent computes its own path and updates the token accordingly.
- *Token Passing with Task Swaps (TPTS)* is a variant of *TP* where a free agent can also choose to be assigned an already assigned task whose execution has not started yet. This approach can improve the efficiency of the

solution with respect to *TP* because agents that are closer to the pickup location of a task can replace another farther agent that has already picked that task.

- *Central* is a centralized algorithm that makes decisions for multiple agents at a time. Just like *TPTS* it considers both free and unexecuted tasks when doing target assignment, but *Central* uses a centralized target-assignment algorithm, the Hungarian method [3]. It uses *CBS* to plan paths for multiple agents.

*TP* and *TPTS* can solve all well-formed MAPD instance. A MAPD instance is well-formed if and only if:

- The number of tasks is finite.
- There are no fewer non-task endpoints than the number of agents.
- For any two endpoints, there exists a path between them that traverses no other endpoints.

However, only *TP* scales efficiently for MAPD instances with a large number of agents and tasks.

*Rolling-Horizon Collision Resolution* [4] has been proposed as an algorithm that decomposes a MAPD instance into a sequence of windowed MAPF instances and replans paths once every $h$ time steps (where $h$ is a user-specified parameter). This algorithm requires to solve collisions only for a limited amount of time steps and it has been shown to scale up to hundreds of agents without considerable drops in the efficiency.

#### 2.2.2 Tasks distribution

The time and space distribution of tasks across the environment can impact the efficiency of an algorithm. It is worth noticing that most of the works about MAPF and MAPD don't run experiments about the effects of different tasks distributions on the objective functions resulting from the proposed algorithms. Many experimental data are the result of tests on environments where the distribution of pickup or delivery locations of tasks is random and uniform [8][15][9]. Thus these results are not always reliable to assess the expected performance of an algorithm in more complex scenarios, for example where pickup and delivery locations are concentrated in some zones of the environment, as it usually happens in real warehouses.
In some works some specific environments have been considered with a more complex distribution of agents' initial position that simulates possible real-world scenarios, like a warehouse [4][7], but the pickup and delivery locations of tasks are still uniformly distributed in most of cases.

Among the aforementioned algorithms, only algorithms that are applied for offline MAPD instances take into consideration the future incoming tasks *before* their arrival time, by taking advantage of the complete knowledge of the sequence of tasks. Other algorithms assume that there is not any useful information available about properties of the incoming tasks until they are added to the task set. This assumption seems to be excessively strict for many real-world scenarios where some information could be retrieved about the probability of a task with a specific location to be added to the system at a certain time step.

### 2.3. Discussion

The area of Multi-Agent Path Finding and Multi-Agent Pickup and Delivery offers new opportunities that aim to make the solutions as efficient as possible in real-world applications.
The main goal of a large part of current research is to fill the gap between the theoretical formulation of MAPF and MAPD problems and their concrete applications. Some of the practical problems that have been faced and integrated in the problem formulations and algorithms for MAPF, such as the robustness to delays and failures that could happen to real robots that execute tasks [1][2], are now being studied for MAPD as well.
Salzman and Stern [10] report as one of the most interesting but not yet explored challenges about MAPD research the analysis of how to exploit the model of the stream of incoming tasks to improve the performance of MAPD

algorithms. This includes both the rate at which tasks arrive and the distribution of pickup and delivery location of each task. Even though the task distribution is not known a priori, *online local adaptation* of the algorithm could be considered, that means learning or estimating the distribution of queries that arrive in an online manner and adapting the system accordingly.

## References


[1] Atzmon, D., Felner, A., Stern, R., Wagner, G., Bartak, R., and Zhou, N. k-robust multi-agent path finding. In *Proceedings of the Symposium on Combinatorial Search (SoCS)* (2017), pp. 157–158.

[2] Atzmon, D. Stern, R., Felner, A., Sturtevant, N., and Koenigz, S. Probabilistic robust multi-agent path findin. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)* (2020), pp. 29–37.

[3] Kuhn, H. W. The hungarian method for the assignment problem. In *Naval Research Logistics Quarterly* (1955), pp. 83–97.

[4] Li, J., Tinka, A., Kiesel, S., Durham, J., Kumar, S., and Koenig, S. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2021), pp. 11272–11281.

[5] Liu, M., Ma, H., Li, J., and Koenig, S. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2019), p. 1152–1160.

[6] Ma, H., Harabor, D., Stuckey, P., Li, J., and Koenig, S. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2019), pp. 7643–7650.

[7] Ma, H., Hoenig, W., Kumar, S., Ayanian, N., and Koenig, S. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2019), pp. 7651–7658.

[8] Ma, H., Li, J., Kumar, S., and Koenig, S. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2017), pp. 837–845.

[9] Nguyen, V., Obermeier, P., Son, T., Schaub, T., and Yeoh, W. Generalized target assignment and path finding using answer set programming. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (2017), pp. 1216–1223.

[10] Salzman, O., and Stern, R. Research challenges and opportunities in multi-agent path finding and multiagent pickup and delivery problems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2020), pp. 662–667.

[11] Sharon, G., Stern, R., Felner, A., and Sturtevant, N. Conflict-based search for optimal multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2012), pp. 563–569.

[12] Sharon, G., Stern, R., Goldenberg, M., and Felner, A. The increasing cost tree search for optimal multi-agent pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2011), pp. 662–667.

[13] Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, S., Boyarski, E., and Bartak., R. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the Symposium on Combinatorial Search (SoCS)* (2019), pp. 151–158.

[14] Surynek, P. An optimization variant of multi-robot path planning is intractable. In *AAAI Conference on Artificial Intelligence* (2010), pp. 1261–1263.

[15] Surynek, P., Felner, A., Stern, R., and Boyarski, E. An empirical comparison of the hardness of multi-agent path finding under the makespan and the sum of costs objectives. In *Proceedings of the Symposium on Combinatorial Search (SoCS)* (2016), pp. 145–147.

[16] Svancara, J., Vlk, M., Stern, R., Atzmon, D., and Bartak, R. Online multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2019), pp. 7732–7739.

[17] Yu, J., and LaValle, S. M. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI Conference on Artificial Intelligence* (2013), pp. 1444–1449.