42+ Flavors of Multi-Agent Path Finding

DAVID ZAHRÁDKA*, Czech Technical University in Prague, Czech Republic
MIROSLAV KULICH, Czech Technical University in Prague, Czech Republic
JIŘÍ ŠVANCARA, Charles University, Czech Republic
ROMAN BARTÁK, Charles University, Czech Republic

Multi-Agent Path Finding (MAPF) is a well-studied problem. In recent years, many derivations of MAPF – MAPF *flavors* – have been formulated and solved. Although some flavors only lift some assumptions present in MAPF, others result from combining two or more flavors. So far, no published work has focused specifically on these newly derived problems, which resulted in incomplete coverage of this area of MAPF, even if it enjoys significant research focus. We present a survey on MAPF flavors, in which we collected 48 existing MAPF-derived problems, categorized them, identified their key defining features, and mapped the relationships between them. The survey is designed to serve as a crossroads for both new and experienced researchers looking to obtain or update their knowledge about MAPF-adjacent problems and readers with such a problem looking for a solution or a reference method. The collected problems are therefore presented in a way that allows an interested reader to find and identify both new and related problems, learn about the state-of-the-art solution methods, and find literature that explains them in greater detail.

JAIR Track: Multi-Agent Path Finding

JAIR Associate Editor: Sven Koenig

JAIR Reference Format:
David Zahrádka, Miroslav Kulich, Jiří Švancara, and Roman Barták. 2026. 42+ Flavors of Multi-Agent Path Finding. *Journal of Artificial Intelligence Research* 86, Article 44 (July 2026), 46 pages. DOI: 10.1613/jair.1.21183

1 Introduction

Multi-Agent Path Finding (MAPF) (Stern et al. 2019) is a problem of finding a set of collision-free paths for a fleet of agents. The agents move in a shared discrete environment with time discretized into timesteps, and each agent has a unique start and goal. The environment can be represented as a grid map or a graph, and it can contain static and dynamic obstacles. The agents are homogeneous, holonomic, and perfectly synchronized, performing exactly one action per time step, all at the same time. In each time step, each agent can move to a neighboring cell (or vertex), or wait in its current place. Both the environment and the positions of all agents are fully known at all times.

Although these assumptions are strong, they are reasonable in some applications. One of the most prominent examples is autonomous warehouses (Wurman et al. 2008), where agents model mobile robots capable of carrying

*Corresponding Author.

Authors' Contact Information: David Zahrádka, david.zahradka@cvut.cz, ORCID: 0000-0002-7380-8495, Czech Institute of Informatics, Robotics and Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic; Miroslav Kulich, ORCID: 0000-0002-2812-8968, kulich@cvut.cz, Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic; Jiří Švancara, ORCID: 0000-0002-6275-6773, svancara@ktiml.mff.cuni.cz, Charles University, Prague, Czech Republic; Roman Barták, ORCID: 0000-0002-6717-8175, bartak@ktiml.mff.cuni.cz, Charles University, Prague, Czech Republic.

DOI: 10.1613/jair.1.21183

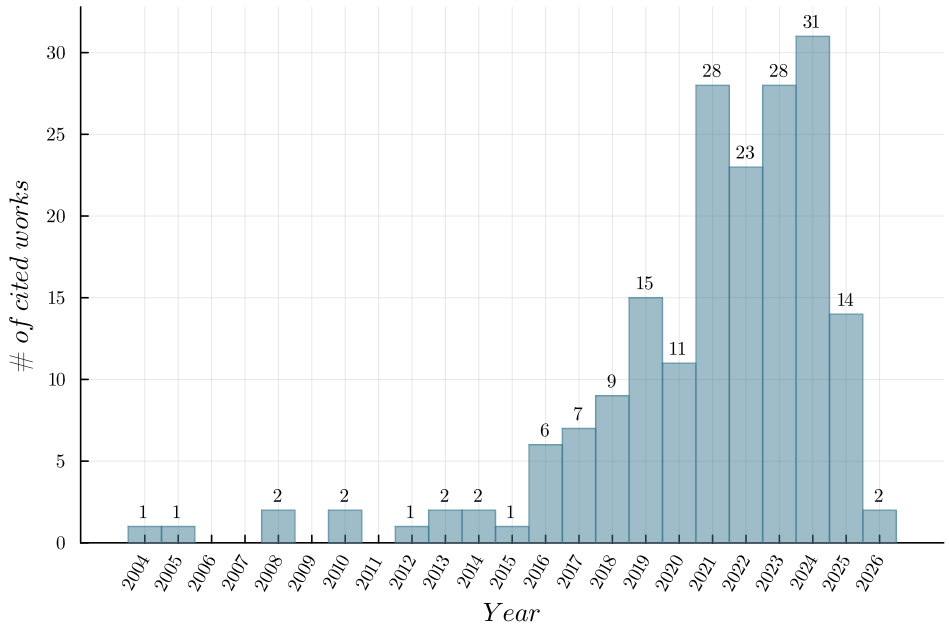

Fig. 1. Publication years of cited works.

items. There, full knowledge of the environment and centralized control necessary for fleet synchronization are possible because the operational space can be purposefully built for the task.

However, for many applications, the assumptions of MAPF may be too strong: the environment may not be discrete, it may not be fully known, or the fleet may not be fully synchronized. An example can be MAPF with continuous space (Surynek 2021a), where the agents' actions may take an arbitrary amount of time. For other applications, MAPF may only be a subproblem to solve as part of a larger, more general problem. An example of this could be Task Assignment and Path Finding (TAPF) (Hönig, Kiesel, et al. 2018), where the goal is to first assign the goals to agents, and then solve MAPF to find collision-free paths. The two problems are intertwined since tasks should be assigned on the basis of the cost of the resulting collision-free paths, but these must be found before task assignment may be evaluated. Therefore, decoupling them may lead to suboptimal solutions.

There is a large body of research focused on dealing with the consequences of lifting the different assumptions of MAPF and how to solve the emerging problems. The resulting family of problems derived from MAPF – MAPF *flavors* – is expansive. Many of the flavors remain very close to MAPF, and there is strong transitivity of solution approaches: Most of the proposed solutions to the flavors are modifications of popular MAPF algorithms. In some cases, it is necessary to use specialized solvers or a combination of multiple methods.

The first goal of this survey is to aggregate different flavors of MAPF and their applications, categorize them, and present their defining features and limitations. The main focus is on recent research that is not captured in already existing surveys, as seen in the distribution of the cited works' publication years in Figure 1. The second goal is to present a high-level overview of the methods developed for each of the presented MAPF flavors so that anyone solving a similar problem can easily find relevant literature or identify the shortcomings of existing approaches. In addition, links to used datasets are provided, if available.

1.1 Existing Surveys

In this Section, we discuss surveys on MAPF that have already been conducted and published. Although all except one focused mainly on classical MAPF, some included sections relating to MAPF variations as well. A basic introduction and an overview of classical MAPF was presented in (Ma and Koenig 2017), which introduces the relevant terms and provides an introduction to solution methods. A more comprehensive overview of classical MAPF, including common objective functions, solution approaches, variations of the problem, and a description of widely used benchmarks, was presented in (Stern et al. 2019). There are two more surveys specialized on solution approaches: one for optimal search-based methods (Felner, Stern, et al. 2021) and one for reduction-based approaches (Surynek 2022). A recent survey focused on solution methods, dividing them into classical and beyond classical (Gao, Y. Li, X. Li, et al. 2024). Furthermore, another recent survey presents an overview of the existing applications of machine learning to MAPF and its related problems (Alkazzi and Okumura 2024).

Finally, a short survey was presented that focuses mainly on the MAPF variations relevant to item transport in autonomous warehouses in (Ma, Koenig, et al. 2017). The authors discuss motivation and applicability of three variants of MAPF that include task assignment and motivate the problem of *robust execution* of MAPF plans. Some MAPF variations and their categories are also described in solution-focused surveys (Gao, Y. Li, X. Li, et al. 2024; Ma 2022; Stern 2019; Stern et al. 2019), such as robust, online, and goal-based variations. For example, (Stern 2019) and (Stern et al. 2019) briefly summarize 7 and 12 different problems, respectively. Although the presented variations are diverse, several important MAPF variations are missing, such as Combinatorial MAPF, MAPF with Precedence Constraints, or Combined Task Assignment and Path Finding. In (Felner, Stern, et al. 2021), only six problems are briefly described, missing many interesting problems such as Multi-Agent Pickup and Delivery (MAPD). The most comprehensive overview of MAPF variations so far is presented in (Ma 2022) and (Gao, Y. Li, X. Li, et al. 2024), which mention 16 and 19 problems, respectively. Being the two newest works, they also include many new methods. However, some important problems are not mentioned in (Ma 2022), such as Multi-Agent Combinatorial Path Finding. Additionally, some problems are not characterized precisely enough, such as Online MAPF and MAPF with Continuous time. Shortcomings also appear in (Gao, Y. Li, X. Li, et al. 2024), which, for example, does not distinguish between distinct problems like TAPF and Anonymous MAPF sufficiently. Although parts of these literature reviews overlap with the topic of our survey, they i) have a smaller scope, ii) do not focus on small distinctions between close variants, and iii) do not discuss the relations between variants that share noticeable similarities. Furthermore, since the most recent related surveys (Gao, Y. Li, X. Li, et al. 2024; Ma 2022), additional works have been published which are included in our survey, as seen in Figure 1.

1.2 Contributions

Since MAPF solution approaches are the subject of numerous expansive survey papers, we focus only on MAPF flavors. These MAPF-related problems have not yet been well summarized, even when they have been the focus of many interesting works in recent years. Researchers focusing on solving new variations of MAPF may therefore have a difficult time finding methods for very similar problems that they could use as a baseline reference method in their work. The main contributions of our survey fill the gaps in the reviews of the existing literature and correct common misconceptions about some of the flavors of MAPF. More concretely, we focus on the following:

(1) Collecting existing MAPF variations in one place, categorizing them and summarizing their defining features.
(2) Presenting a high-level overview of the existing methods to solve the problems.
(3) Mapping the relationships between different MAPF flavors, their similarities and their differences.

The survey is designed as a catalog of MAPF flavors that can be used by the reader to quickly orient themselves in the vast field of MAPF-derived problems. Then, for each problem the reader is interested in, it is easy to

find whether a problem with some specific characteristics was solved, get a brief overview of the methods and optimization criteria used, and access existing relevant research work to learn more.

First, in Section 2, we present a definition of MAPF and briefly describe the commonly used solution approaches and the applications of MAPF. Section 3 then contains a brief overview of the categories in which we divide the flavors. Each category is then further explained in its own Section (Sections 4 to 8), together with a description of each of its flavors and their state-of-the-art. All categories are introduced by a table that contains a brief overview of the defining features of each of the flavors before presenting each flavor in more detail. Afterwards, unique problems that do not fit into any category are presented in Section 9. Finally, concluding remarks and interesting statistics are presented in Section 10.

2 MAPF

The MAPF problem is defined by a graph $G = (V, E)$, where $v \in V$ are vertices, $e \in E$ are edges, and by a set of agents A. Each agent $a \in A$ starts at a unique starting location $v_s^a \in V$ and has a unique goal location $v_g^a \in V$ it needs to reach. Time is discretized into timesteps t starting at zero and ending at infinity, with T_0^∞ representing the entire available time. In each timestep t, each agent a can perform an action represented by a pair $\alpha_a = (v, v')$, representing movement over an edge $e = (v, v')|(v, v') \in E$. If $v = v'$, the agent is *waiting* in v, and if $v \neq v'$, the agent moves into a neighboring vertex v'. The agents can move to any neighboring vertex. The agents are also assumed to be *perfectly synchronized*, which means that they always execute their actions at the same time and that the actions take exactly one time step. An ordered sequence of actions $\pi(a) = (\alpha_a^1, \alpha_a^2, \ldots, \alpha_a^n)$ is called a path, with n being its length.

The goal is to find a set of collision-free paths $\Pi = (\pi(a^1), \pi(a^2), \ldots, \pi(a^m))$, where m is the number of agents, such that each agent starts at its starting vertex v_s^a and ends at its goal vertex v_g^a. After reaching its goal, an agent usually remains there, occupying the vertex. Alternatively, agents may disappear after reaching their goals, which is a less common definition. There are multiple types of conflicts (collisions) (Stern 2019). The first is the *vertex* conflict that occurs when two or more agents occupy the same vertex at the same time. The second type of conflict is the *edge* conflict, in which two agents move over the same edge in the same direction at the same time. This conflict is always preceded and followed by a vertex conflict, since the agents must be in the same vertex in the first place and will arrive in the same vertex again. The third type is the *swap* conflict, in which two agents traverse the same edge in the *opposite* directions. In other words, the agents swap their positions in a single timestep. The fourth type of conflict is the *following* conflict, which occurs when an agent moves into a vertex at the same timestep that another agent leaves it. The last type of conflict is the *cycle* conflict, a specific type of following conflict that occurs when there is a group of agents moving in a circular pattern following each other. The cycle conflict is the only type that always involves more than two agents: a cycle conflict with only two agents is the swap conflict.

The vast majority of papers on MAPF and its derivations forbid only vertex and swap conflicts. Edge conflicts are usually not considered because all edge conflicts are resolved by resolving vertex conflicts. The last two conflict types – following and cycle – are also frequently allowed, since they do not actually result in collisions between the agents. However, recognizing following and cycle conflicts is important when dealing with imperfect execution, since following and cycle movement can only be safely executed when the fleet is perfectly synchronized. This is relevant, for example, in the problem of robust execution of MAPF plans (described later in Section 6.7), in which some methods cannot execute plans containing cycle conflicts.

Most works focus on minimizing one of two objective functions: *flowtime*, also known as sum of costs (SOC) or sum of individual path costs (SIC), and *makespan*. Flowtime is equal to the sum of the lengths of all agents' paths, or, in other words, the sum of times when each agent reaches its goal. Makespan is equal to the time that the whole task is completed, i.e., the time the last agent reaches its goal or the maximum of individual path costs. A

notable difference is that with makespan, an agent that is not the last to reach its goal can move around without incurring additional costs. Flowtime, on the other hand, minimizes the time that all agents are still on their way, which can represent fuel or battery charge. Optimizing MAPF is an NP-hard problem (Surynek 2010). What makes finding optimal flowtime and makespan solutions for MAPF on 2D grids difficult was also studied (Geft 2023). A study on the differences between the optimization of makespan and flowtime was presented in (Švancara, Atzmon, et al. 2024).

2.1 Common Solution Approaches

Methods for MAPF can be divided either by the solution quality (optimality) and scalability that they offer or by the algorithmic principles that they use. Regarding solution quality and scalability, there are three classes of methods for MAPF:

(1) Optimal, which provide the best solutions but have difficulties scaling for large numbers of agents.
(2) Bounded-suboptimal, which can find solutions guaranteed to be within a user-defined optimality boundary, offering the option to trade quality guarantees for better scaling or speed.
(3) Suboptimal, which have no guarantees on solution optimality and sometimes are not even guaranteed to be complete, but scale well and are fast.

Independently of these three classes, the methods can be further divided by their algorithmic approach into two main categories: reduction-based approaches and search-based approaches. Both reduction-based and search-based approaches exist for all three solution quality classes. Reduction-based approaches work by reducing MAPF into a decision problem, such as the Boolean Satisfiability problem (SAT) or Network Flow (J. Yu and LaValle 2013), while search-based approaches find solutions by using algorithms such as A* for planning paths for individual agents coupled with various higher-level frameworks that resolve collisions. Reduction-based approaches scale worse with increasing map size, whereas search-based approaches struggle with an increasing number of collisions, due, for example, to a high density of the instance or a large number of agents (Švancara, Atzmon, et al. 2024; Švancara and Barták 2019).

The current state-of-the-art reduction-based solvers model the position of each agent in time as a decision variable (Achá et al. 2022; Surynek 2019; Surynek, Felner, et al. 2016), which means that it is required to specify the number of timesteps in the expected solution. If there is no such solution within the given number of timesteps, additional timesteps are included. Constraints on the position of the agents (such as correct transitions and no conflicts) are expressed in the given formalism. Since the size of the generated instance (for example, in terms of the number of Boolean variables) is the limiting factor in solving the problem, some approaches trying to limit the size have been developed (Husár et al. 2022). Still, reduction-based approaches tend to scale worse on large maps while being able to tackle small, densely populated instances. A significant advantage of the reduction-based approaches is that they leverage powerful off-the-shelf solvers. Any new development in these solvers is therefore easy to utilize in MAPF. Reduction-based solvers exist in all three classes: optimal (Achá et al. 2022), bounded-suboptimal, and suboptimal (Surynek, Felner, et al. 2018).

Perhaps the most prominent representative of search-based methods for MAPF and its flavors is Conflict-Based Search (CBS) (Sharon et al. 2012). It is a two-level optimal algorithm: the lower level is responsible for finding paths for the agents, while the higher level resolves collisions by constraining the low-level planner. For this purpose, CBS builds a constraint tree that contains the intermediate solutions (potentially containing collisions), their cost, and a set of constraints for each agent. Optimality is guaranteed by always exploring the intermediate solution with the lowest cost. It is precisely because of this constraint tree that algorithms such as CBS have trouble scaling with an increasing number of collisions: each resolved collision branches the constraint tree, and some collisions have to be resolved repeatedly over many different branches. Recent improvements to CBS try

to limit the excessive branching by detecting such cases and creating constraints capable of preventing many collisions at once (J. Li, D. Harabor, et al. 2021).

CBS can be modified to a bounded-suboptimal version, called Enhanced CBS (ECBS) (Barer et al. 2014). There are also suboptimal search-based approaches, such as Prioritized Planning (Berg and Overmars 2005), and from recent works, metaheuristic solvers such as Large Neighborhood Search (J. Li, Z. Chen, D. Harabor, et al. 2022). Prioritized planners solve MAPF by assigning or finding a priority ordering to the agents and then planning their paths one by one, treating the higher-priority agents as dynamic obstacles. Metaheuristic solvers iteratively refine the solution by modifying its parts using various strategies and are able to find near-optimal solutions. They often use fast suboptimal MAPF solvers to quickly generate new solutions.

Although search-based suboptimal methods scale very well, with Large Neighborhood Search being able to plan for up to 1000 agents, they struggle with numbers that are even higher. However, there are other suboptimal methods, such as rule-based solvers, that are capable of solving these complex instances, even though they often produce solutions with larger costs. Examples of these methods can be the rule-based Priority Inheritance via Backtracking (PIBT) (Okumura, Machida, et al. 2022) or hybrid methods, such as Lazy Constraint Addition Search for MAPF (Okumura 2024), which uses PIBT as a generator of different configurations of agents coupled with a search procedure over the different configurations. Other methods that are neither reduction-based nor search-based are, for example, learning-based methods that have been developed in recent years, such as PRIMAL (Damani et al. 2021) or RAILGUN (Tang, X. Xiong, et al. 2025). These methods also show promising scalability at the cost of solution quality.

Since methods for MAPF are not the focus of this survey, this list is not complete. A more complete overview can be found in existing surveys that focus on MAPF solution methods (Felner, Stern, et al. 2021; Gao, Y. Li, X. Li, et al. 2024; Ma 2022).

2.2 Applications

Perhaps the most natural application for MAPF[1] is collision-free path planning for mobile robotic fleets in autonomous warehouses and production lines (Lehoux-Lebacque et al. 2024; Wurman et al. 2008). MAPF algorithms were also evaluated on maps representing 3D warehouses (Q. Wang et al. 2024). Another problem in which MAPF was used is for a team of mobile robots performing agricultural tasks, such as harvesting and transporting produce (Jo and Son 2024). Another natural application is in video games to plan the movement of characters (Ma, Yang, et al. 2017; Sigurdson, Bulitko, Yeoh, et al. 2018). Video game maps are also very often used as benchmarks for MAPF algorithms (Stern et al. 2019).

Path planning for Unmanned Aerial Vehicle (UAV) swarms is another area where MAPF was successfully applied. In one approach, a 3D roadmap was constructed and by solving MAPF on the roadmap, collision-free paths for a UAV swarm were obtained (Hönig, Preiss, et al. 2018). A similar problem was formulated in (Hönig, Satish Kumar, et al. 2016), which dealt with a mixed group of robots changing formations. Each robot belongs to a different group, and they are required to maintain a formation, while not having a prescribed goal position for each single one of them. Instead, a robot belonging to a specific group is required to move to a specific location. The same problem but with a different motivation was presented in (Barták and Mestek 2021), where groups of robots were assigned different colors, and the goal was to form an image by moving any robot with a specific color to a specific position. A slightly different problem is UAV traffic management (Ho, Geraldes, et al. 2022; Ho, Goncalves, et al. 2019), where the goal is to plan paths for the UAVs to get to their goals (for example, delivery locations) and back. The UAVs can share start and goal locations, but in the case of a shared start, they must have different start times. In addition, UAVs may have different speed limits and sizes. A different application of MAPF

[1]Applying MAPF to a practical problem often requires adding specific constraints, technically making it a MAPF flavor. In this section only, we use MAPF as a collective label for MAPF and its flavors.

to UAVs is presented in (Choudhury et al. 2021), where MAPF is used to find collision-free paths for a drone swarm. The drones have limited range, but can ride on public transport vehicles to extend it.

A popular application for MAPF is to plan the movement of trains. MAPF algorithms were used in (J. Li, Z. Chen, Zheng, et al. 2021) to win an international competition in train routing, where MAPF outperformed reinforcement learning-based methods. Other works applying MAPF for trains include (Atzmon, Diei, et al. 2019) and (Švancara and Barták 2023), which extend the problem to trains of variable length. A related problem is the application of MAPF in convoy movement planning (Thomas et al. 2015), where convoys occupy edges that represent roads. The convoys can be longer than an edge, occupying multiple different edges at the same time, or, in the case of short convoys, fit multiple of them on the same edge. MAPF was also used to formulate the problem of path planning for general swarms in emergency situations, such as evacuation (Janovská and Surynek 2021). Here, agents were split into two categories: leaders, which had full information and their behavior was controlled by a MAPF solver, and followers, which were acting based on local rules, such as following an assigned leader, following the nearest leader, and others.

MAPF was also applied to traffic and car-related problems. One application was trajectory planning for cars, such as valet parking (Okoso, Otaki, Koide, et al. 2022; Okoso, Otaki, and Nishi 2019). Another is autonomous intersection management, presented in (Švancara, Vlk, et al. 2019), where an intersection manager, as described in (Dresner and Stone 2008), handles vehicles arriving at an intersection. Each vehicle requests a collision-free trajectory to its desired lane, which the manager provides, and the vehicle must then follow. In (Morris et al. 2016), an application was presented in which MAPF was used to model a problem of planning for autonomous towing vehicles at airports that are tasked with taxiing airplanes between gates and runways.

MAPF was also used for the virtual network embedding problem to optimize the allocation of limited resources provided by the physical infrastructure for virtual networks (Zheng, Ravi, Kline, Koenig, et al. 2022). An improved version was later presented (Zheng, Ravi, Kline, Thurlow, et al. 2023), and the problem was also solved with a reduction-based approach (Surynek, Zheng, et al. 2024). In (Belov et al. 2021), MAPF is applied to 3D pipe routing, where the goal is to connect equipment with pipes in a 3D environment with static obstacles. There are many additional constraints, such as specific bending radii for the pipes and maximum distance of the pipes from other equipment to minimize support costs. MAPF was also successfully applied to collaborative robotic manipulators in bin picking scenarios (Shaoul et al. 2024).

3 MAPF Flavors

In this Section, we group the assumptions of MAPF, presented in Section 2, into different categories. Then, we explain how MAPF flavors emerge by lifting some of these constraints, and we present an overview of the categories of MAPF flavors. Finally, we explain the way each flavor will be described in Sections 4 to 9.

The assumptions of MAPF can be separated into five main categories.

(1) The first category covers the assumptions about the goals of the problem. This includes the assumptions that each agent has one given unique goal, that the goal is one vertex, and that the agent can reach its goal at any time the goal is not occupied by another agent.
(2) The second category includes the assumption that all information about the problem is available *a priori*, making MAPF a so-called *instantaneous assignment* problem (Gerkey and Matarić 2004). For example, the full set of agents, their starting and goal locations, and the locations of all obstacles are known at the beginning.
(3) The third category is the assumptions about *certainty*, such as perfect synchronization of the fleet and perfect execution of all actions.

(4) The fourth category is the assumptions about the environment. This includes that the agents operate in a discrete environment with discretized time, and that the environment is homogeneous and that different vertices do not have varying qualities.
(5) The fifth (and last) category is the assumptions about the agent models. These include the assumptions that the agents are holonomic, occupy one vertex at a time, and do not have any kinematic or dynamic constraints. Typically, it is assumed that the agents move in a two-dimensional space.

The majority of MAPF flavors emerge by removing or modifying one of these assumptions. Most commonly, MAPF flavors are derived by generalizing the rules for agents' *goals*. For example, each agent may not be tied to a specific destination, but instead, assigning the destinations to the agents may be a necessary subproblem. In other cases, agents may be assigned more than one goal, or the goals may have multiple *subgoals* that need to be visited by the assigned agent. In some problems, the agents may be required to cooperate to fulfill individual goals. The goal-based variations of MAPF are described in Section 4.

Sometimes not all information is available at the beginning. Using the terminology proposed in (Gerkey and Matarić 2004), these problems feature *time-extended assignment* rather than instantaneous. For example, as the execution of the plan progresses, new goals for the agents may appear or new agents might be added to the problem, and the algorithm has to also include them in the solution. This might require replanning. Alternatively, additional obstacles may appear during execution of a plan. Such problems are called MAPF flavors with time-extended assignment and are summarized in Section 5.

By adding uncertainty to MAPF, it is possible to express that, for some information, only an estimate is available. However, the solution must remain feasible for most if not all possible outcomes. Uncertainty in MAPF is most frequently associated with properties of actions. For example, adding upper and lower bounds on travel time to an edge may represent that it is not certain how long it will take the agent to traverse the edge. Alternatively, there may be some probability that an agent fails to execute an action on time. The category of MAPF flavors with action uncertainty is presented in Section 6.

In classical MAPF, agents move in a shared environment that is represented by a graph. The agents occupy the vertices and move over the edges. Typically, each vertex has up to four neighbors, one for each cardinal direction (up, down, left, right) and one self-loop, indicating that an agent may wait in the vertex. This graph typically represents a 2D grid map, with cells represented by vertices. However, it is possible to formulate MAPF for an environment that has, e.g., continuous space or time. This may require changing the way collisions are detected and resolved or adding sampling procedures to plan movement in continuous space. MAPF flavors in special environments are described in Section 7.

MAPF with special agent model is a category for problems where the agents behave in a different way than in original MAPF. An example of a special agent model is a nonholonomic car-like agent, or an agent that occupies multiple vertices at the same time. There is a seemingly large overlap between this category and MAPF with special environments, mostly because the behavior of nontypical agents can be encoded into the underlying graph that represents the environment. An example can be resolving the nonholonomicity of a vehicle by using vertices to represent not only the possible locations of the agents, but also their configurations including heading. However, generating such a graph may be a problem due to the large number of possible configurations. Another example can be UAVs that move in a discrete 3D space. Although a discrete three-dimensional environment can be easily represented with a graph, it may be necessary to deal with other problems, such as limited flight time, or downwash when one UAV flies directly above another. For this reason, we consider such cases as MAPF with special agent models, but necessitating a special environment as well. MAPF flavors with special agent models are presented in Section 8.

Some MAPF-related problems are too unique to be included in the aforementioned categories, such as the problem of selecting the best algorithm to solve a MAPF instance, optimizing multiple objectives at the same

time, or requiring agents to stay close together. The problems that cannot be included in any other category are collectively presented in Section 9.

Each Section describing a category of flavors starts with an explanation of the different features which define the flavors in the category. An overview of the features of each flavor is presented in the form of a table. The columns, representing individual features, are ordered according to their impact on the problem. For example, the presence of an additional optimization problem (such as Task Assignment) is more impactful than changing a property of the agents' goals. The problems are ordered according to the features they contain, both in the table and in the text that follows. This means that first, we list problems without Task Assignment before moving on to problems where it is present. In addition, the tables contain information on whether the flavor was solved with one of the two most common approaches – a reduction-based approach or a CBS-based method – or whether it was solved with other methods, such as metaheuristic or rule-based solvers, learning-based approaches, search-based methods other than CBS or methods that are not applicable to classic MAPF.

Each problem description in Sections 4-9 starts with a brief single-sentence summary of the flavor and a list of closely related problems. An example can be seen here:

MAPF with some new features.
Related problems: Some Similar Problem (-missing feature, +added feature, ⋄different in feature)

These are included to help the reader quickly gain an insight into what differentiates the flavor both from classical MAPF and from other similar problems. Each related problem contains a summary of differences listed as keywords preceded by one of three symbols:

(1) Minus (−) indicates that the related problem lacks a feature.
(2) Plus (+) indicates that the related problem has a feature that does not exist in the current flavor.
(3) Diamond (⋄) indicates that both the related problem and the current flavor have a common feature, but with important differences.

Since the list of related problems aims to be comprehensive, it often includes flavors that are defined later in the survey, sometimes in a different category altogether. Therefore, the list of related problems best serves the reader who has perused the different categories and their problems already. The related problems always link to each other for easy navigation. A visualization of the relationships between different flavors is available online[2].

Then, we characterize the flavor, discuss its potential similarities and differences compared to other problems, and summarize the methods that were used to solve the problem and which optimization criteria were utilized.

4 Goal-Based Variations

Goal-based MAPF variations change the number or nature of goals that the agents must reach. In this Section, we first introduce common characteristics of problems belonging to this category. Then we list the flavors and their characteristics in Table 1, before continuing with a description of each problem and existing solutions. The problems are presented in the order they appear in Table 1.

The first characteristic of the goal-based flavors is the presence of *Task Assignment* (TA). This means that in some flavors, the goals are not initially assigned to the agents, but it is a subproblem that must be solved. TA may be formulated, for example, as a form of a Travelling Salesman Problem or a Vehicle Routing Problem. If a problem features TA, there may be different degrees of freedom as to which agent can be assigned which goal. This is called *anonymity*, and can be either *full* or *selective*. In fully anonymous problems, any goal can be assigned to any agent. In problems with selective anonymity, it is possible for each goal to define a subset

[2]Diagram of the relationships between different flavors of MAPF: https://imr.ciirc.cvut.cz/Datasets/Mapf-flavors

Table 1. Different features of goal-based variations of MAPF. TA stands for Task Assignment; MG for multigoal; SG for subgoals, and Anon. for anonymity. The Method column indicates which methods were used to solve the problem: R appears if a reduction-based approach was used, C if a CBS-based approach was used and O if some other type of method was used.

Problem	Section	TA	MG	SG	Destination	Anon.	Goal constraints	Method
MAPF	2	×	×	1	Last goal	N/A	×	RCO
MAPF-DL	4.1	×	×	1	Last goal	N/A	G-deadline	RC
MAPF-DT	4.2	×	×	1	Last goal	N/A	Tardiness penalty	R
MG-MAPF	4.3	×	×	N	Last goal	N/A	×	RC
OMG-MAPF	4.4	×	×	N	Last goal	N/A	Order	C
MAPF-PC	4.5	×	×	N	Last goal	N/A	S-precedence	CO
AMAPF	4.6	✓	×	1	Last goal	Full	×	RO
AMAPFwID	4.7	✓	×	1	Last goal	Full	I-deadline	R
Co-MAPF	4.8	✓	×	2	None	Full	Cooperation	C
MG-TAPF	4.9	✓	×	N	Last goal	Full	Order	C
TAPF	4.10	✓	✓	1	Last goal	Selective	×	RC
MSMP	4.11	✓	✓	1	Any distinct	Full	×	O
MCPF	4.12	✓	✓	1	Distinct subset	Selective	×	CO
OMAPD	4.13	✓	✓	2	None	Full	Order	RCO
PC-TAPF	4.14	✓	✓	2	None	Full	G-precedence, Order	C
OMAPD-TD	4.15	✓	✓	2	None	Full	Order, I-deadline	O
GTAPF	4.16	✓	✓	N	Last goal	Selective	I-deadline	R

of agents it can be assigned to. Note that this is a generalization of full anonymity: the subset may contain all agents, making the problem fully anonymous.

Some problems are *multigoal*: this means that the agent might have to visit more than one goal. Most often, each goal consists of a single *subgoal*, which is the location the agent must visit to fulfill the goal. Sometimes, there are two subgoals, representing *pickup* and *delivery* of an item, and in some cases there is an arbitrary number N of possible subgoals or actions for each goal.

The distinction between a goal and a subgoal is important for problems with TA: Two different goals may be assigned to two different agents, while one goal with two subgoals must always be serviced by the same agent. Furthermore, a goal is not completed until all its subgoals are completed. This is important in problems with *deadlines*, where completing a goal versus completing a subgoal would affect the achieved score.

Another important feature of the problems is the *destination* of the agents. The destination refers to a position that the agent must reach to end its path and remain there. In classical MAPF, the destination is the *single* goal that the agent is assigned. There are more options when the agent has multiple goals. Sometimes, the agent can end in any one of its goals. Other times, it might be required to end in a distinct location, which is not part of the goals. This occurs only when TA is involved, because in such a case, we can freely reassign the goals (possibly each with multiple subgoals) between the agents, but each agent still has its defined destination location. The agent may not even be required to end anywhere specific, and can move wherever it is most convenient after finishing its tasks. We recognize four different destination types: *last goal*, where the agent's destination is the last goal it visits, *any distinct*, where the agent can end in any unoccupied destination location, separate from the goals, *distinct subset*, where the agent can end in a subset of destinations (again excluding the goals), and *none*, where the agent can freely roam after finishing its goals. Note that if the agent has only one goal, it is also its last goal.

The last distinguishing feature is whether there are any constraints imposed on the goals. The agent may be required to visit its multiple goals in a specific order, or the goals may have a deadline by which they must be

visited. A *G-deadline* indicates that there is a global deadline by which the agents need to reach the goals. *I-deadline* means that each goal has its own deadline, while *tardiness penalty* merely penalizes agents that arrive too late. *Precedence* constraints between the goals or subgoals mean that some goals or subgoals must be completed before others. Precedence constraints between goals are indicated as *G-precedence* and between subgoals as *S-precedence*. If a goal has multiple subgoals, they are sometimes *ordered*: this is a special type of S-precedence, which means that the agent has to visit the subgoals in a specific order to complete the goal. Order constraints are generally specified for all subgoals of all goals. And finally, *cooperation* means that multiple agents must work together to complete individual goals, in contrast to each agent being fully capable of completing its own goals.

Note that the problems in this category still assume instantaneous assignment, that is, all information is available at the beginning. Thus, problems where new goals appear over time are not included and are instead listed in the category of MAPF with time-extended assignment (Section 5).

4.1 MAPF with Deadlines (MAPF-DL)

MAPF with a global deadline.
Related problems: MAPF-DT (⋄deadlines); GTAPF (+multigoal TA, +subgoals, ⋄deadlines); AMAPFwID (+TA, ⋄deadlines)

In MAPF-DL, there is a deadline T_{dl} that places a hard constraint on the maximum makespan of the solution (Ma, Wagner, et al. 2018). With this deadline, it may happen that not all agents can feasibly reach their goals. Therefore, the main goal is to maximize the number of agents that reach their goals within the time imposed. This criterion can also be expressed as a variant of sum of costs (flowtime), where the cost of an agent that did not reach its goal is one, while the cost of an agent that reached its goal is zero. The problem was solved optimally using a modification of CBS (Ma, Wagner, et al. 2018), and also using a reduction to Network Flows (Ma, Wagner, et al. 2018) and Constraint Satisfaction Problem (J. Wang et al. 2019).

4.2 MAPF with Due Times (MAPF-DT)

MAPF minimizing tardiness.
Related problems: MAPF-DL (⋄deadlines); AMAPFwID (+TA, ⋄deadlines)

MAPF-DT, also called Multi-Robot Path Planning with Due Times (MRPP-DT), is a generalization of MAPF-DL in which, instead of a global deadline, each agent has its own *due time* (H. Wang and W. Chen 2022). Due time expresses the latest time that an agent should reach the goal. However, in contrast to MAPF-DL, the evaluation is not necessarily binary. If an agent arrives later, the difference between its arrival time and the due time is expressed by the so-called *tardiness*. Three different optimization criteria that can be minimized were presented for the problem: i) maximum tardiness, ii) total tardiness, and iii) total unit penalties. Maximum tardiness is similar to makespan and total tardiness to flowtime. If the third criterion, total unit penalties, is used and all agents have the same due time, the problem becomes MAPF-DL. The authors of (H. Wang and W. Chen 2022) also formulate a fully anonymous version of the problem (AMRPP-DT), where the goals are not tied to agents. The problem was solved for these custom criteria with a reduction-based approach by reducing the problem into Integer Linear Programming.

4.3 Multi-Goal MAPF (MG-MAPF)

MAPF with unordered subgoals.
Related problems: MAPF-PC (+precedence); OMG-MAPF (+order); MG-TAPF (+TA, +order)

In MG-MAPF, each agent has a set of subgoals which it must visit instead of a single goal (Surynek 2021b). While the problem is called Multi-Goal, in the terminology used in this survey the nature of each agent's goals is more akin to subgoals since they are closely tied with the specific agent. The agent can visit the subgoals in any order, and, at the end, it remains in the last visited subgoal. Therefore, the problem is twofold: i) optimizing the order in which each agent visits its subgoals and ii) finding collision-free paths such that the agents visit their subgoals in the determined order.

The problem was solved in the original work using both a search-based approach and a reduction-based approach (Surynek 2021b). The search-based approach was decoupled: first, it found the order in which the agent visits each subgoal, and then found a set of collision-free paths for each agent using CBS. The reduction-based approach was integrated and utilized Satisfiability Modulo Theories (SMT). Both methods focused on minimizing flowtime. Although the CBS-based method is presented as optimal, its low-level planner was shown to be suboptimal (Mouratidis et al. 2024).

4.4 Ordered Multi-Goal MAPF (OMG-MAPF)

MAPF with ordered subgoals.
Related problems: *MG-MAPF (-order); MAPF-PC (⋄precedence); MG-TAPF (+TA); MG-MAPD (+lifelong, +TA, +multigoal)*

In OMG-MAPF, similarly to MG-MAPF, each agent has a set of subgoals that it must visit (Mouratidis et al. 2024). In contrast to MG-MAPF, the problem is simplified in that the subgoals already have a predefined order in which they must be visited. The problem is therefore only to find a set of collision-free paths that satisfies the subgoal ordering. OMG-MAPF is part of a decoupled MG-MAPF solution procedure, where the first step is to order the subgoals, and the second step is to solve the resulting OMG-MAPF for the ordered set of subgoals (Surynek 2021b). At the end of their paths, the agents remain in their last subgoal. The problem was solved with two different modifications of CBS for both flowtime and makespan (Mouratidis et al. 2024; Surynek 2021b).

4.5 MAPF with Precedence Constraints (MAPF-PC)

MAPF with precedence-constrained but unordered subgoals.
Related problems: *OMG-MAPF (⋄precedence); MG-MAPF (-precedence); PC-TAPF (+multigoal TA)*

MAPF-PC is a generalization of OMG-MAPF that introduces *precedence constraints* to the subgoals (H. Zhang et al. 2022). Precedence constraints between a pair of subgoals indicate that one of the subgoals can be completed only after the other subgoal has been completed. In contrast to OMG-MAPF, subgoals belonging to different agents can be precedence-constrained. The agents may complete the subgoal later than they first visit its location, and therefore agents may wait in the subgoal's location before completing it if necessary. Each subgoal can have multiple precedence constraints. The problem was solved with an optimal CBS-based method and a suboptimal method, both optimizing flowtime (H. Zhang et al. 2022).

4.6 Anonymous MAPF (AMAPF)

MAPF with fully anonymous goals.
Related problems: *AMAPFwID (+deadlines); TAPF (+multigoal, ⋄anonymity); MG-TAPF (+subgoals, +order); MSMP (+multigoal, ⋄destination); OTIMAPP (+uncertainty)*

In AMAPF – also called unlabeled MAPF – the goals are not assigned to agents, and any agent can end its path at any goal, provided it is unique (J. Yu and LaValle 2013). In other words, the solution is feasible if all goals are

occupied, each by some unique agent. AMAPF and its derivations have an interesting property that they are most often optimally solved by reduction to Network Flow problem and they have only polynomial complexity for makespan optimization (J. Yu and LaValle 2013). Recently, new improvements speeding up the Network Flow approach were introduced in (Ali and Yakovlev 2024), and a decentralized method for flowtime and makespan was also introduced (Dergachev and Yakovlev 2024).

4.7 AMAPF with Individual Deadlines (AMAPFwID)

MAPF with fully anonymous goals and individual deadlines.
Related problems: *AMAPF (-deadlines); MAPF-DT (-TA, ⋄deadlines); MAPF-DL (-TA, ⋄deadlines); GTAPF (+multigoal, +subgoals, ⋄anonymity); OMAPD-TD (+multigoal, +subgoals, ⋄destination)*

AMAPFwID is a generalization of AMAPF where each anonymous goal has a *deadline*, by which it must be *acquired* (Fine et al. 2023). A goal is acquired if there is an agent occupying it. The deadlines are different from the global deadline in MAPF-DL (Section 4.1) in which it is also tolerated that some goals are not completed: the number of goals completed is an optimization criterion. The authors introduce three distinct variants of the problem: i) agents disappear upon reaching a target, ii) agents stay on targets, and iii) agents can move away from targets after their deadline expires as long as the goal stays occupied. In the first case, all agents located on a goal at its deadline disappear from the environment. In the second case, the agents remain occupying the goal locations, but cannot move, blocking them for other agents. In the third case, the goal location must remain acquired after its deadline, but not necessarily by the same agent. The authors introduce the *hotswap* action, in which an agent may move into a goal location acquired by another agent while the acquiring agent moves out of it. The case where agents remain on their targets is close to MAPF-DT with so-called total unit penalties (Section 4.2). AMAPFwID was solved using reduction to Network Flow focused on minimizing traveled distance (flowtime without wait actions).

4.8 Cooperative MAPF (Co-MAPF)

MAPF where completing a goal (task) involves two agents: one visits the starting location of the task, then meets with the other agent in some vertex to transfer it so that the second agent can complete it by visiting the goal location of the task.
Related problems: *CF-MAM (subproblem); OMAPD (-cooperation, +multigoal)*

In Co-MAPF, the agents have to *cooperate* to complete goals (Greshler et al. 2021). The agents are divided into two categories of the same size: *initiators* and *executors*. Each task has to be assigned to a unique pair consisting of an initiator and an executor. There is exactly one task for each pair of agents. To complete the task, the pair of agents must perform a sequence of three actions: first, the initiator must visit the task's starting location. Then, the two agents must meet by visiting any vertex at the same time (this is allowed as an exception to vertex collisions). Finally, the executor must visit the goal location to complete the task. Therefore, each task consists of two subgoals, like OMAPD (Section 4.13). The subproblem of finding a meeting place and navigating the agents there is a special case of the Conflict-Free Multi-agent Meeting problem (Section 9.3). Co-MAPF was solved with a modification of CBS for flowtime, and the authors claim that the results can be generalized for makespan. A suboptimal CBS-based solver for flowtime and makespan with better scaling was presented in (Gao, Y. Li, Mu, et al. 2025).

4.9 Multi-Goal Target Assignment and Path Finding (MG-TAPF)

MAPF with fully anonymous TA and subgoals.
Related problems: *OMG-MAPF (-TA); MG-MAPF (-TA, -order); AMAPF (-subgoals, -order); GTAPF (+multigoal, +deadlines, ⋄anonymity); TAPF (-subgoals, -order, +multigoal, ⋄anonymity); MSMP (-subgoals, -order, +multigoal, ⋄anonymity, ⋄destination); MG-MAPD (+lifelong, +multigoal)*

MG-TAPF is a generalization of AMAPF where each goal consists of an ordered sequence of subgoals (Zhong et al. 2022). If an agent is assigned a sequence of subgoals, it must visit all of the subgoals in the order they are given. At the end of its path, the agent remains in the last goal. The problem is a combination of AMAPF and OMG-MAPF, with the latter problem formulating the path finding subproblem after solving the task assignment. The problem was solved using optimal and bounded-suboptimal modifications of CBS with flowtime as the optimization criterion (Zhong et al. 2022). The paper also includes proof that the problem is NP-hard.

4.10 Combined Target Assignment and Path Finding (TAPF) and Colored MAPF

MAPF with selectively anonymous multigoal TA (TAPF) / single-goal TA (Colored MAPF).
Related problems: *MCPF (⋄destination); MSMP (⋄anonymity, ⋄destination); AMAPF (-multigoal, ⋄anonymity); MG-TAPF (-multigoal, +subgoals, +order, ⋄anonymity); GTAPF (+subgoals, +deadlines); OMAPD (+order, +subgoals, ⋄anonymity, ⋄destination)*

TAPF is a generalization of AMAPF in which the agents are separated into *teams*, and each team has a specific set of goals (Ma and Koenig 2016). The goal is to find a set of collision-free paths such that every agent ends in a unique goal that is assigned to the agent's team. Assigning the goals to the agents is therefore an important part of the problem.

In the original definition of TAPF, there is exactly one goal for each agent (Ma and Koenig 2016). This version of TAPF is also known as Colored MAPF (Barták, Ivanová, et al. 2021a,b). However, TAPF was later used to describe a more general problem in which each team can have a different number of goals than agents (Hönig, Kiesel, et al. 2018). Since most works focus on the multigoal variant and the original problem is also defined by another name, we use TAPF exclusively for the latter multigoal flavor. The problem generalizes both MAPF and AMAPF: for N agents, 1 team and a single goal per agent, the problem becomes AMAPF. Conversely, for N agents, N teams but still with a single goal per agent, the problem becomes the classical MAPF.

TAPF was solved using CBS-based optimal and bounded-suboptimal methods to optimize flowtime (Hönig, Kiesel, et al. 2018). The flowtime-optimal method was further improved in (Tang, Z. Ren, et al. 2023), and the bounded-suboptimal version was also improved in (Tang, Koenig, et al. 2024).

Colored MAPF was solved with a reduction-based method (Barták, Ivanová, et al. 2021b) and with a CBS-based method that used reduction to Network Flow (Ma and Koenig 2016), all for makespan. A different criterion is *idle time* of sortation centers, located at the agents' goals, which was minimized by a Network Flow-based method (Kou et al. 2020). The problem differs from typical Colored MAPF because each station may be assigned to multiple agents. The problem was also extended in (Kou et al. 2020) into a so-called Lifelong version, in which not all goals are known at the start (Section 5.2).

4.11 Multi-Agent Simultaneous Multi-goal Sequencing and Path Finding (MSMP)

MAPF with multigoal TA where agents collect goals on the way to their destinations (full anonymity).
Related problems: *MCPF (⋄anonymity, ⋄destination); TAPF (⋄anonymity, ⋄destination); AMAPF (-multigoal, ⋄destination); MG-TAPF (-multigoal, +subgoals, +order, ⋄anonymity, ⋄destination)*

MSMP is a problem similar to AMAPF (Section 4.6): there is a set of fully anonymous destination locations that the agents must reach, one for each agent (Z. Ren, Rathinam, et al. 2021a). However, in MSMP, the agents have an additional task in the form of goals that the agents must visit on their way to their destinations. Each goal must be visited at least once and the goals are also fully anonymous, meaning that any agent can visit any goal. This is reminiscent of MG-TAPF (Section 4.9), where each goal consists of multiple subgoals. However, since the goals in MSMP may be visited independently of each other, MSMP is a multigoal problem rather than a problem with subgoals. An agent can also visit any number of goals. Therefore, MSMP features the additional task assignment subproblem to decide which agent visits each goal and which agent ends at each destination location. In (Zahrádka, Andreychuk, et al. 2022), a similar problem is solved with two notable differences: Each agent has a predefined unique destination and the agents have a maximum number of goals that each can visit, modeling the limited carrying capacity of mobile robots. The authors call the problem Multi-Agent Multi-Item Pickup and Delivery (MAMPD).

MSMP was optimally solved by combining a method for the Multiple Traveling Salesman Problem with a method for MAPF, using flowtime as a criterion (Z. Ren, Rathinam, et al. 2021a). The related MAMPD problem was solved using a suboptimal metaheuristic, but the authors present a lower bound estimation method and show that the solutions are near-optimal (Zahrádka, Andreychuk, et al. 2022). The optimized criteria are flowtime and Total Travel Delay, which is a criterion that measures throughput (Z. Chen, Alonso-Mora, et al. 2021). The authors provided the code publicly[3] together with instances[4].

4.12 Multi-Agent Combinatorial Path Finding (MCPF)

MAPF with multigoal TA where agents collect goals on the way to their destinations (selective anonymity).
Related problems: TAPF (⋄destination); GTAPF (+subgoals, +deadlines); MSMP (⋄anonymity, ⋄destination)

MCPF (Z. Ren, Rathinam, et al. 2022) is another problem formulation where the agents have to visit goals on their way to their destinations, similar to MSMP. In contrast to MSMP, each destination and goal may be assigned to a specific subset of agents. MCPF is, therefore, a more general version of MSMP. If the subset contains all agents, the destination or goal becomes fully anonymous. Additionally, if there are no goals and each destination is assigned to exactly one agent, the problem becomes the classical MAPF.

A variant of this problem is MCPF with Heterogeneous Task Duration (MCPF-D) (Y. Zhang, Wu, et al. 2024), in which the agents need to stay at the task (goal) location for some time to complete it. Each task may have a different duration.

MCPF was solved using a modification of CBS for flowtime (Z. Ren, Rathinam, et al. 2022). The solver is publicly available[5]. A bounded-suboptimal variant exists (Z. Ren, Rathinam, et al. 2024), and also a solver originally developed for MSMP (Z. Ren, Nandy, et al. 2024) that optimized makespan. MCPF-D was solved using a variation of CBS minimizing the sum of completion times (Y. Zhang, Wu, et al. 2024).

4.13 Offline Multi-agent Pickup and Delivery (OMAPD)

MAPF with multigoal TA where each goal has pickup and delivery locations.
Related problems: PC-TAPF (+precedence); OMAPD-TD (+deadlines); Co-MAPF (-multigoal, +cooperation); TAPF (-order, -subgoals, ⋄anonymity, ⋄destination); MAPD (+lifelong); tMAPD (+lifelong, ⋄subgoals); N-MAPD (+gen. environment, ⋄traversability)

[3] MAMPD solver: https://github.com/zahrada2/mampd_decoupled_gaps
[4] MAMPD instances: https://github.com/zahrada2/mampd_instances
[5] MCPF solver: https://github.com/rap-lab-org/public_pymcpf

OMAPD (Liu et al. 2019), also called Combined TAPF (Henkel et al. 2019), is similar to TAPF, since each agent can be assigned more than one goal. The main difference between OMAPD and TAPF is that i) the tasks are fully anonymous, ii) the agents do not have a destination (meaning they can move around after finishing all their tasks), and iii) each task consists of exactly two actions: *pickup* and *delivery*. The actions are ordered, and thus an agent must first visit the pickup location of a task before visiting its delivery location. The agent can complete only one task at a time, which means that it cannot collect an item and then pick up a second one before delivering the first one.

The problem was solved for makespan using a suboptimal prioritized algorithm (Liu et al. 2019), with an optimal modification of CBS that minimized the duration of task fulfillment (Henkel et al. 2019), and using Ant Colony optimization with travel distance as a criterion (Qizilbash et al. 2020).

A very close problem is Double-Deck MAPD (DD-MAPD) (B. Li and Ma 2023), also called Multi-Agent Transportation (MAT) (Bachor et al. 2023) or Multi-Agent Warehouse Rearrangement (MAWR) (Sherma et al. 2025), in which the items are the obstacles (shelves) that the agents can carry. The problem is a combination of OMAPD and Terraforming MAPD (Section 5.7), because picking and placing shelves changes the environment: an agent carrying a shelf cannot move under another shelf. Each shelf has a goal location, but agents may temporarily place the shelves elsewhere. DD-MAPD was solved for makespan and flowtime with a framework that assigns shelves with the Hungarian algorithm and then uses pre-computed paths to deliver the shelves while avoiding collisions with free agents (B. Li and Ma 2023). The problem was also solved by decomposing the problem into a sequence of SAT problems (Bachor et al. 2023). In (Sherma et al. 2025), it was solved optimally for makespan with a CBS and network flow-based algorithm that repeats two phases: in the first phase, it uses CBS to plan the movement of shelves. In the second phase, it uses a Network Flow algorithm to find paths for the agents such that they fulfill the plan from the first phase.

4.14 TAPF with Precedence Constraints (PC-TAPF)

MAPF with multigoal TA and subgoals and with precedence constraints between goals.
Related problems: OMAPD (-precedence); MAPF-PC (-multigoal TA)

PC-TAPF adds precedence constraints between goals (Brown et al. 2020) to OMAPD. The problem models an assembly plant in which each goal has two ordered subgoals, representing an item's pickup and delivery locations. It also models higher-level tasks, such as assembly tasks, that have preconditions (necessary items that must be placed near a machine) and operation duration after which a new item is ready to be transported somewhere else.

PC-TAPF was solved by a CBS-based algorithm for makespan (Brown et al. 2020). However, as explained in (H. Zhang et al. 2022), the CBS used in (Brown et al. 2020) may produce suboptimal solutions due to its nature of handling paths with multiple subgoals by chaining together individual segments between subgoals. The same problem was later investigated in (Mouratidis et al. 2024), a paper focusing on OMG-MAPF (Section 4.4). The focus of (H. Zhang et al. 2022) is on MAPF-PC (Section 4.5), which is the path-planning subproblem of PC-TAPF.

4.15 OMAPD with Task Deadlines (OMAPD-TD)

MAPF with multigoal TA where each goal has pickup and delivery locations and a deadline.
Related problems: OMAPD (-deadlines); GTAPF (-order, ⋄subgoals, ⋄anonymity, ⋄destination);
AMAPFwID (-multigoal, -subgoals, ⋄destination); MAPD (+lifelong, -deadlines)

A variation of OMAPD is OMAPD with Task Deadlines (OMAPD-TD) (Wu et al. 2022). Here, each task has an individual deadline and the goal is to maximize the number of completed tasks. Therefore, not all tasks must be

completed for the solution to be feasible, and completion is an optimization objective. Thus, the problem is at an intersection of OMAPD, MAPF-DL (Section 4.1) and AMAPFwID (Section 4.7). Like in OMAPD, there may be more than one task per agent. Therefore, it has the added decision problem of deciding which task to complete and which to forego during TA. The problem was solved with a suboptimal search-based decoupled method.

4.16 Generalized TAPF (GTAPF)

MAPF with selectively anonymous multigoal TA, subgoals, and deadlines.
Related problems: *TAPF (-subgoals, -deadlines); MCPF (-subgoals, -deadlines); MG-TAPF (-multigoal, -deadlines, ⋄anonymity); AMAPFwID (-multigoal, -subgoals, ⋄anonymity); MAPF-DL (-multigoal TA, -subgoals, ⋄deadlines); OMAPD-TD (+order, ⋄subgoals, ⋄anonymity, ⋄destination)*

GTAPF is a combination of TAPF and MG-TAPF in which each task (goal) contains multiple subgoals like in MG-TAPF, each agent can be assigned multiple tasks like in TAPF, but in addition, each task is also subject to a deadline (Nguyen et al. 2017). The task's deadline represents the time by which all of its subgoals must be completed, similarly to the individual deadlines in AMAPFwID (Section 4.7). The problem formulation also allows one to introduce order constraints on the subgoals, among many other modifications. However, the problem solved in (Nguyen et al. 2017) did not consider the additional modifications. The goal is to fulfill each task while minimizing some criterion, such as flowtime or makespan. GTAPF has been solved both optimally and suboptimally using Answer Set Programming with makespan as the optimization criterion, but the authors mention that the method works for other criteria as well.

5 MAPF with Time-Extended Assignment

Some MAPF flavors lift the *full a priori information* assumption. This means that some parts of the problem are not known at the start and that the information appears while the solution is being executed. This property of problems is known as time-extended assignment (Gerkey and Matarić 2004). These generalizations are often application-inspired, such as by autonomous warehouses and production plants, where the full task of the robotic fleet is not known at the beginning. For example, transport of some additional items may be requested while the fleet is still fulfilling the previous request. Not only is it necessary to incorporate the newly added goals into the solution, but also the current solution may need to be updated. This may require *replanning*.

There are three different areas in which new information may arrive over time: i) the tasks, ii) the agents, and iii) the environment. Varying tasks means that not all objectives are known at the beginning and they may appear later. Problems with varying tasks are also known as *lifelong* problems (Ma, J. Li, et al. 2017). Varying agents means that we do not know the full set of agents (including their starting locations and goals) at the beginning, and problems with varying environment may feature either an environment partially unknown at the start or a changing one. Furthermore, these problems often have some unique feature that describes the specific dynamic of the problem. The problems are classified in Table 2.

5.1 Dynamic MAPF (DMAPF)/Generalized DMAPF (GDMAPF) and Online MAPF

MAPF where new agents or obstacles may appear over time and their goals may change.
Related problems: *tMAPD (⋄var. tasks, ⋄var. environment); LMAPF (-var. agents, -var. env, ⋄var. tasks); MAPFOU (-var. tasks, -var. agents, ⋄var. environment); MTPF (+special agent model, ⋄varying agents)*

There are two different problems known as DMAPF, one a generalized version of the other. We denote them as DMAPF and Generalized DMAPF (GDMAPF). DMAPF, studied in (Wan et al. 2018), is a MAPF flavor in which new agents *appear* during execution. Each agent has its assigned goal. After an agent reaches its goal, it returns

Table 2. Different features of MAPF flavors with time-extended assignment.

Problem	Section	Varying Environment	Varying Agents	Varying Tasks	Feature	Method
DMAPF	5.1	×	✓	×	Agents appear & disappear	RCO
LMAPF	5.2	×	×	✓	Task = 1 goal	CO
MAPD	5.3	×	×	✓	Task = 2 goals	O
CMAPD	5.4	×	×	✓	≥ 1 item simultaneously	O
MG-MAPD	5.5	×	×	✓	Task ≥ 2 goals	O
MAPFOU	5.6	✓	×	×	Some obstacles unknown	CO
tMAPD	5.7	✓	×	✓	Obstacles are items	O
GDMAPF	5.1	✓	✓	✓	Agents & obstacles appear, goals may change	R

to its starting position and leaves the map. This behavior is inspired by caching areas where agents wait for tasks. The cost of each agent's path is equal to the number of steps it takes before it reaches the goal. Therefore, the return trip does not increase it. GDMAPF is a more general problem (Atiq et al. 2020), in which not only new agents appear, but also new obstacles can appear. In addition, the goals of the agents may change during execution. DMAPF was solved optimally with a modification of CBS for the flowtime criterion (Wan et al. 2018). GDMAPF was solved using Answer Set Programming with makespan as the criterion.

A very close problem to DMAPF is Online MAPF (Švancara, Vlk, et al. 2019), in which new agents also appear with time, but the agents disappear upon reaching their goals instead of returning to their start positions. This represents the agents moving into some private space, such as a private garage. The agents can also appear anywhere on the map, reflecting the situation where they are entering from a private parking space they were waiting in. Multiple solution strategies based on both CBS and reduction were presented for the sum of costs objective (Švancara, Vlk, et al. 2019). These methods used replanning to deal with newly arriving agents. They also argue that the makespan criterion is impractical in this case, since it becomes infinite when new agents keep appearing indefinitely. While Online MAPF and DMAPF are very similar problems and both were solved by CBS, the solvers are noticeably different. For example, CBS for Online MAPF minimizes replanning time by being able to replan only for a subset of agents when a new agent appears, while CBS for DMAPF reuses existing constraints to achieve a similar result.

Furthermore, a variation of Online MAPF where the agents arrive periodically is studied in (Kasaura et al. 2023). It was solved with a two-phase suboptimal method with custom-designed criteria. The problem also involved continuous space, which will be discussed later in Section 7.4.

5.2 Lifelong MAPF (LMAPF)

MAPF where the agents are assigned new goals as they complete them.
Related problems: *MAPD (+TA, +subgoals); GDMAPF (+var. agents, +var. env, ⋄var. tasks); CC-PP (+positioning requirements)*

In LMAPF, the full set of goals is not initially known, and agents are asking for new goals from some task assigner as they complete the current goals (J. Li, Tinka, et al. 2021). Each agent has a maximum of one assigned goal at a time. The task assigner is not considered part of the solution.

The problem was solved using a suboptimal Receding (Rolling) Horizon Collision Resolution algorithm (RHCR) (J. Li, Tinka, et al. 2021) with *throughput* (number of goals completed per time step) as a criterion. The main idea behind RHCR is to decompose LMAPF into a sequence of so-called Windowed MAPF problems, where collisions are always resolved only for a fixed number of timesteps into the future – a fixed-length *window* (*horizon*) that

always starts at the current time step. RHCR was also used with machine learning (Madar et al. 2022), and behaviors in case of failure to find a solution were studied (Morag, Stern, et al. 2023). In (Jiang, Y. Zhang, et al. 2024), the authors discuss challenges related to LMAPF with Rotations (discussed later in Section 8.1), a problem used in the League of Robot Runners competition (Chan et al. 2024). There is also a method[6] that iteratively improves the solution while it is being executed (Y. Zhang, Z. Chen, et al. 2024), and a suboptimal method that focuses on avoiding congestion (Z. Chen, D. Harabor, et al. 2024). LMAPF was also solved with an imitation learning-based method (Jiang, Y. Wang, et al. 2024). A decentralized learning-based approach was investigated in (Skrynnik et al. 2023).

A related problem is MAPF for Lifelong (MAPF4L) (Morag, Gabay, et al. 2025), also called MAPF with Reachability Objective (MAPFRO) (Felner and Stern 2026), which was formulated as a subproblem of LMAPF in dense environments. Since agents in LMAPF receive new goals as they complete them, it is not necessary for all agents to be in their goals at the same time. Therefore, MAPF does not accurately model the subproblem of planning paths for the current TA by requiring it. Instead, a more accurate formulation is MAPF4L, in which it is sufficient that every agent visits its goal at some point in its path. This means that an agent can reach its goal but then leave it to make room for another agent on its way to a goal. In MAPF4L, both agents would be counted as having completed their paths successfully. Although MAPF4L is a so-called one-shot problem (meaning it is not a time-extended assignment problem), it is closely tied to this problem family. The problem was solved with an optimal modification of CBS, an LNS-based metaheuristic solver, and two other suboptimal methods for the throughput criterion (Morag, Gabay, et al. 2025).

Another closely related problem is MAPF with Unassigned Agents (MAPFUA) (Felner and Stern 2026), in which some agents do not have a goal, but can be freely moved around. MAPFUA models an LMAPF subproblem in which the TA is already completed, but there were fewer available goals than agents. The unassigned agents still occupy their vertices, but they can move to free a path for an agent that has an assigned goal. Three different optimization criteria were proposed: sum of service time, which is the flowtime of assigned agents, fuel, which is the total flowtime without counting wait actions, and number of unassigned agents that move, in which the number of unassigned agents that are required to perform an action is minimized. The problem was solved for the flowtime of assigned agents with CBS and for the number of unassigned agents that move criterion with a CBS using a modified A* in its low level (Felner and Stern 2026).

5.3 Multi-Agent Pickup and Delivery (MAPD)

MAPF with multigoal TA where each goal has pickup and delivery locations and new tasks arrive over time.
Related problems: *LMAPF (-TA, -subgoals); CMAPD (+capacity); MG-MAPD (+subgoals); OMAPD (-lifelong); OMAPD-TD (-lifelong, +deadlines); N-MAPD (-lifelong, +gen. environment); tMAPD (+var. environment, ⋄subgoals)*

MAPD is a problem directly inspired by automated warehouses, where a fleet of robots is tasked to transport items around the warehouse (Ma, J. Li, et al. 2017). Similar to OMAPD (Section 4.13), each item has a *pickup location* and needs to be delivered to its *delivery location*. Therefore, there are two subgoals per each item, and each agent can carry one item at a time. Unlike OMAPD, new items appear over time, much like in LMAPF (Section 5.2). However, in contrast to LMAPF, the problem involves task assignment to decide which agent will pick up and deliver which item and in what order. This problem can therefore be seen as a combination of OMAPD and LMAPF. Since each item has two subgoals and each agent can be assigned more than one item, the path finding problem can be formulated as OMG-MAPF (Section 4.4). An offline version of this problem was also studied (OMAPD, Section 4.13). A Token Passing method for the service time criterion, which is the average

[6]LMAPF solver with iterative improvement: https://github.com/YueZhang-studyuse/PIE

completion time for tasks, was presented (Ma, J. Li, et al. 2017). An optimal search-based algorithm for flowtime is also available (Lam et al. 2024).

A variant of MAPD with additional locations called *caches*, known as LMAPF with Cache Mechanism (LMAPF-CM), is presented in (Tang, Z. Yu, et al. 2025). As in MAPD, each item has a pickup location and a delivery location. Each item belongs to a specific type and each item type is located on a unique shelf. The delivery locations are shared, called *unloading ports*. The agents do not immediately transport the items from the shelves to the unloading ports, but first transport the shelf with the requested item type to a cache area near the unloading ports and only then retrieve an item from the shelf to deliver it. Since there are more shelves than cache locations, it is necessary to prevent situations where a shelf that is still required for item delivery is swapped for another one. It was solved with a combination of a custom task assigner and LaCAM (Okumura 2023) and the solver is publicly available[7]. The tasks were generated based on real data about item demand[8].

5.4 Capacitated MAPD (CMAPD)

MAPF with multigoal TA where each goal has pickup and delivery locations, new tasks arrive over time, and each agent can carry multiple items at the same time.
Related problems: MAPD (-capacity)

In CMAPD, each agent may carry more than one item at the same time (Z. Chen, Alonso-Mora, et al. 2021). This requires adding another subproblem to the task assignment process. In contrast to MAPD, where each delivery action directly follows a pickup action, in Capacitated MAPD, it is necessary to decide their exact order in such a way that the capacity constraint is not violated. If the agents have capacity for only a single item, the problem becomes MAPD. Each task has a *release time*, which is the time at which it becomes available to pick up. It is also the time at which it becomes known to the algorithm.

The problem was solved using a suboptimal LNS-based metaheuristic optimization method (Z. Chen, Alonso-Mora, et al. 2021). It optimizes a specially tailored criterion, Total Travel Delay, which measures throughput by counting how much later each item is delivered compared to its earliest possible delivery time. It is computed by taking the delivery time of each item and subtracting its release time and the length of the shortest path from its pickup location to the delivery location. The method and instances are publicly available[9].

5.5 Multi-Goal MAPD (MG-MAPD)

MAPF with multigoal TA where new goals arrive over time and each may have multiple ordered subgoals.
Related problems: MAPD (-subgoals); OMG-MAPF (-lifelong, -TA, -multigoal); MG-TAPF (-lifelong, -multigoal)

MG-MAPD is a generalization of MAPD in which items can have different numbers of ordered subgoals, rather than exactly two (Xu et al. 2022), similar to OMG-MAPF (Section 4.4) or MG-TAPF (Section 4.9). Each task can consist of an arbitrary number of actions that are locations that the agent must visit. The actions are ordered and, to complete an assigned task, an agent must visit the associated location of each action in the order they are given. As is the case with MAPD, not all tasks are known at the beginning. The problem was solved with a suboptimal LNS-based metaheuristic that minimized *service time*, which is the time that the tasks are waiting to be completed.

[7] LMAPF-CM solver: https://github.com/HarukiMoriarty/CAL-MAPF
[8] Item demand data: https://kaggle.com/datasets/felixzhao/productdemandforecasting
[9] CMAPD solver and instances: https://github.com/nobodyczcz/MCA-RMCA

5.6 MAPF with Obstacle Uncertainty (MAPFOU)

MAPF where some obstacles are unknown at the beginning and can be detected only when an agent is near it.
Related problems: tMAPD (+var. tasks, ⋄var. environment); GDMAPF (+var. tasks, +var. agents, ⋄var. environment)

In MAPFOU, the assumption that the entire environment is fully observable at all times is lifted, and some static obstacles might be unknown (Shofer et al. 2023). An example of the known obstacles can be the walls, whereas the unknown obstacles might be items placed on the ground, or closed doors. These obstacles can be detected by the agents using on-board sensors. Such sensors have a limited range, and thus agents can only detect the obstacle when they are sufficiently near. In (Shofer et al. 2023), it is assumed that the agent can sense the obstacle only when it is right next to it. Although the problem's name implies uncertainty, the fact that the obstacles become known after they are discovered makes it a problem with time-extended assignment.

The authors investigated two different scenarios: one with centralized execution, which allows for replanning and in which the agents share their observations, and one with decentralized execution, where the agents do not share their observations. The goal was to minimize the best- and worst-case flowtimes. Both versions were optimally solved, with the decentralized version solved with a modification of CBS.

Another version of the problem is MAPD with External Agents (MAPD-EA), in which the goal is to solve MAPD (Section 5.3) among uncontrollable agents, acting as dynamic obstacles with unknown paths (Bonalumi et al. 2025). The problem was also called MAPF with Dynamic Uncontrollable Agents (MAPDUA) (Strawn et al. 2025). Part of the problem is learning how dynamic obstacles move. The problem was solved with a Token Passing method for service time, and for a finite set of tasks, the problem can be solved for makespan (Bonalumi et al. 2025). A CBS-based approach that also learned how the agents move and was combined with a Receding Horizon approach (Section 5.2) for adjusting paths during the execution was presented in (Strawn et al. 2025).

5.7 Terraforming MAPD (tMAPD)

MAPF where the goals are shelves with pickup and delivery locations and also act as obstacles.
Related problems: GDMAPF (⋄var. tasks, ⋄var. environment); MAPD (-var. environment, ⋄subgoals); DD-MAPD (-lifelong, ⋄subgoals); MAPFOU (-var. tasks, ⋄var. environment)

In tMAPD, the goal is again to carry some items from their pickup locations to their delivery locations, with the difference that the items are a subset of obstacles on the map (Vainshtein et al. 2023). Afterward, the items must be returned to their pickup locations. Furthermore, agents who are not carrying an item (obstacle) are free to pick up obstacles and move them around, for example, to free some passageway. This process is called *terraforming*. Due to the increased complexity of the problem, the terraforming is performed only when a *disturbance* occurs (Vainshtein et al. 2023). That might be a fallen item, or a broken agent, that blocks a passage that would otherwise be free. The nature and time of the disturbances are not known until they occur. The problem was solved by extending the Receding (Rolling) Horizon approach (J. Li, Tinka, et al. 2021) used for LMAPF (Section 5.2), in which the planned paths must be collision-free only for a fixed number of future timesteps, with service time as optimization criterion (Vainshtein et al. 2023).

6 MAPF with Action Uncertainty

A very important category of MAPF flavors is problems that add *uncertainty* into the agents' action execution capabilities. These problems are very often motivated by applications on real robots. One of the key assumptions of classical MAPF is that the whole robotic fleet is perfectly synchronized. However, in practice, this is rarely true, due to limited hardware capabilities of the robots, imperfect localization, etc. The most commonly used uncertainty is *uncertain travel time*, where it is not accurately known how long it will take each robot to execute

each action. Since MAPF assumes that each action takes exactly one time step, this uncertainty may lead not only to suboptimal solutions, but also to a *failure* of the execution, where some robots may become stuck, or the robots may even collide.

One option to deal with the uncertainties is to plan in a way that the plan itself takes them into account, for example, by leaving more space between agents. A different approach is to use the *robust execution* methods. These methods form a middle step between a planner and a robotic fleet. They take a feasible MAPF plan as input and are responsible for communicating it to the robotic fleet in a way that even if a robot is delayed, the execution remains safe.

Table 3. Different features of MAPF flavors with action uncertainty.

Problem	Section	Uncertain Agents	Uncertain Environment	Uncertainty Type	Notes	Method
MAPF-TU	6.1	×	✓	Bounded	Edge travel times	RCO
k-robust	6.2	✓	×	Bounded	Maximum delay k	CO
p-robust	6.3	✓	×	Probabilistic	Safety threshold p	CO
MAPF-DP	6.4	✓	×	Probabilistic	Probabilistic criterion	C
OTIMAPP	6.5	✓	×	Bounded	Arbitrary delay	CO
MAPPCF	6.6	✓	×	Bounded	Agents may break	O
Execution	6.7	✓	✓	Any	Agents execute imperfectly	RO

Table 3 presents three main features of the problems: uncertainty related to agents, uncertainty related to the environment, and the type of uncertainty: whether it is bounded or probabilistic. Uncertainty related to agents means that an agent's action has an uncertain property or outcome. For example, it may be that the time it takes an agent to execute an action is not fixed or that it is not guaranteed that it will be able to perform an action in the desired time step.

Uncertainty in the environment means that a property of the underlying graph or the environment itself cannot be exactly known. This can mean, for example, that it is not certain how long it takes to traverse an edge. There are two main types of uncertainty: bounded and probabilistic. A *bounded uncertainty* can be exactly enumerated. An example of a bounded uncertainty may be the maximum number of timesteps by which an agent may be delayed. When the uncertainty is bounded, the evaluation is simplified since we can verify that a desired property of the solution, such as safety, holds for all values within the bounds.

On the other hand, a *probabilistic (unbounded) uncertainty* cannot be enumerated. For example, the time it takes an agent to execute an action can be expressed as some probability distribution. In such a case, it is not possible to simply enumerate all possible action durations. Since there are many ways uncertainty can be defined in a problem, we also include another column (Notes) in which we add a brief summary of the nature of the uncertainty.

6.1 MAPF with Time Uncertainty (MAPF-TU)

MAPF with upper and lower bounds of travel time for each edge.
Related problems: *k-robust (-unc. environment, +unc. agents); MAPF-DP (-safety guarantee, -unc. environment, +unc. agents); $MAPF_R$ (-uncertainty, ⋄edge costs)*

In MAPF-TU, there is uncertainty as to how long it takes for an agent to move over some edges (Shahar et al. 2021). Each edge in the graph representing the environment has a *lower* and *upper* bound on travel time. This means that, for example, a longer interval between the lower and upper bounds may be assigned to edges in hard-to-traverse areas because it is more difficult to predict how long it will take an agent to traverse them. The

goal of MAPF-TU is to find a solution such that no matter how long exactly each movement will take, all agents still reach their destinations without colliding. The solution of MAPF-TU is safe to execute as long as the lower and upper bounds on the edges' travel times hold.

The authors present multiple settings. The first is the offline setting, for which the authors present two modified optimal algorithms. One is based on A* with Operator Decomposition (Standley 2010) and the other is based on CBS with modified low-level search and collision detection. The criteria for the algorithms were the optimistic flowtime (the sum of the lower bounds of the agents' actions) and pessimistic flowtime (the sum of the upper bounds). No matter the criterion, the solution is only feasible if it remains safe for any combination of action durations.

Another approach for the offline setting was presented in (Švancara, Zahrádka, et al. 2025), where a reduction-based method was used to generate *policies* that guarantee safety instead of plans. The benefit of using a policy is that the agent may take different actions depending on how long it takes the agent to traverse the edges, assuming that the agent knows how long it took and also knows its own location. The method presented in (Švancara, Zahrádka, et al. 2025) was also able to optimize makespan.

Two online settings were also presented in (Shahar et al. 2021), in which replanning is possible. In the first setting, the agents know their positions and the times at which they arrive and can replan their own paths. The agents can use the newly sensed information only to reduce the time they would wait, but cannot deviate from their paths due to the lack of communication. This is because without communication, there is no mechanism to ensure that the potential new paths would be collision-free. The redundant waiting actions can be found by finding a new safe shortest path using only the same vertices as the original path. In the second, the agents can communicate this information between themselves and a central planner can be used to replan the paths of multiple agents. The central planner was the CBS for the offline setting with the same criteria: optimistic and pessimistic flowtimes.

6.2 k-Robust MAPF

MAPF where each agent may be delayed by up to k timesteps.
Related problems: *p-robust (⋄uncertainty type); MAPF-TU (-unc. agents, +unc. environment); LA-MAPF (-uncertainty, ⋄collisions); MTPF (-uncertainty, ⋄collisions); OTIMAPP (+unbounded uncertainty); MAPPCF (⋄delay type, ⋄alternate paths)*

The goal of k-robust MAPF is to produce plans resistant to agents' delays of up to k timesteps (Atzmon, Stern, Felner, Wagner, et al. 2018). The solution is a plan that is safe to execute without any robust execution policy, as long as no agent is more than k actions behind any other agent. In practice, this means that it is assumed that every agent occupies not only its current vertex but also all vertices it occupied during its last k actions. k-robust MAPF is similar to MAPF-TU, which also produces plans that are safe to execute as long as the bounds on delays are not exceeded. However, in k-robust MAPF, the uncertainty in travel time is a property of the agents only, while in MAPF-TU it is a property of the edges. Therefore, in k-robust MAPF, agents may not avoid large uncertainties by avoiding some edges, and there is no reason to differentiate between optimistic and pessimistic criteria, since k is the same for every action. An interesting feature of k-robust MAPF is that a 1-robust solution is a MAPF solution which does not contain following or cycle conflicts.

A modification of CBS that minimized flowtime was presented as a solution to k-robust MAPF (Atzmon, Stern, Felner, Wagner, et al. 2018). Later, a more advanced version of CBS that uses symmetry breaking techniques was presented (Z. Chen, D. D. Harabor, et al. 2021). Another approach to solving k-robust MAPF is to prepare alternative paths for agents that they can use in case a delayed agent would cause a collision. First, the parts of the plan (vertices) where a collision would occur if one of the agents was delayed by up to k timesteps are identified. Then, for every such vertex, an alternate collision-free path is found for the delayed agent with A*.

During execution, if an agent is delayed in such a way that it would cause a collision, it may avoid the collision by opting for the already prepared alternate path. This approach was used with a modification of the standard CBS for makespan (Nekvinda and Barták 2021). A k-robust MAPD problem (Section 5.3) was solved in (Lodigiani et al. 2023) using a Token Passing method and makespan as a criterion.

6.3 p-Robust MAPF

MAPF with probabilistic delays where the goal is to find a solution that is safe with probability at least p.
Related problems: k-robust (⋄uncertainty type); MAPF-DP (-safety guarantee)

p-robust MAPF (Atzmon, Stern, Felner, Sturtevant, et al. 2020) assumes that at each step, an agent may be delayed with some probability. An agent getting delayed at some time step means that for the given time step, the agent did not execute an action. The goal of p-robust MAPF is to find a solution where the probability of safe execution is greater than or equal to some $p : 0 \leq p \leq 1$. The probability of delay of an action is a parameter $p_d : 0 \leq p_d \leq 1$. In (Atzmon, Stern, Felner, Sturtevant, et al. 2020), two algorithms based on CBS are presented – one optimal and one suboptimal – which focused on minimizing flowtime. Furthermore, the authors present two methods to evaluate the p-robustness of the solution inside the algorithms, as it is itself a non-trivial problem. Finally, the authors investigated the percentage of simulations where no conflicts occurred. A p-robust MAPD problem (Section 5.3) was presented and solved in (Lodigiani et al. 2023) using a Token Passing method.

Another similar problem was formulated by (Peltzer et al. 2021), called Stochastic Travel Time MAPF, where the delays were not modeled as an agent failing to execute an action, but rather as noise that is added to edge travel times. The travel times can be nonunit, and the problem formulation also allows different travel speeds for each robot. The goal is to find a solution that is collision-free with probability $1 - \epsilon$, where ϵ is the maximum allowed probability that a collision occurs during execution. The optimization criterion is the expected flowtime of the execution and it was solved using a variation of CBS.

6.4 MAPF with Delay Probabilities (MAPF-DP)

MAPF with probabilistic delays where safety is ensured during execution and the goal is to minimize a probabilistic criterion.
Related problems: p-robust (+safety guarantee); MAPF-TU (+safety guarantee, +unc. environment, -unc. agents)

MAPF-DP, similar to p-robust MAPF, models uncertainty during plan execution as the probability that a robot fails to execute an action (Ma, T. K. S. Kumar, et al. 2017). This means that there is some probability that the robot will not start moving after receiving a *move* action from its plan. If a failure occurs, the agent will attempt to start the action in the next timestep. The goal of MAPF-DP is to minimize *average expected makespan*, which is the average time to execution of the plan after it was affected by delays (Ma, T. K. S. Kumar, et al. 2017). In contrast to p- and k-robust MAPF, the plan is not itself resistant to delays, but only minimizes the execution length taking into account the delay probability. A solution to MAPF-DP may be feasible even if it includes high collision risks if reducing these risks would increase the expected makespan instead of decreasing it. Therefore, the plan must be executed using robust execution policies.

The problem was solved using a suboptimal CBS-based method which optimized the average expected makespan by taking into account how the probability of a failed action propagates further into the agent's plan (Ma, T. K. S. Kumar, et al. 2017). The method is suboptimal because the measured criterion relies on later execution, and the method merely estimates it. The authors also describe the two robust execution policies utilized (Section 6.7).

6.5 Offline Time-Independent Multiagent Path Planning (OTIMAPP)

MAPF where agents must reach their goals no matter how finitely delayed they become.
Related problems: Execution (⋄arbitrary delay); k-robust (-unbounded uncertainty); AMAPF (-uncertainty); MAPPCF (+alternate paths, ⋄unbounded uncertainty)

So far, all problem formulations with time-based uncertainty required some information about the amount of uncertainty, or its prediction at a minimum. In OTIMAPP, such information is not available (Okumura, Bonnet, et al. 2023). Instead, the goal is to find a solution in which, no matter how much each agent is delayed, all of them will reach their destinations as long as they move eventually. The agents have to drive according to their original plans, but may decide to wait until some future moment in time. For this purpose, (Okumura, Bonnet, et al. 2023) studies the possible *deadlocks* that may exist in MAPF solutions that should be avoided in this problem. A method based on Prioritized Planning and a CBS-based method for the makespan criterion are presented (Okumura, Bonnet, et al. 2023). The solvers and a visualizer are available online[10].

OTIMAPP was also extended with task assignment (Okumura and Défago 2023), producing a problem that combines OTIMAPP and AMAPF (Section 4.6). There may be at most the same number of goals as agents, and the goals are fully anonymous. The problem was solved with a suboptimal method for both makespan and flowtime, and the code is publicly available[11].

6.6 Multi-Agent Path Planning with Crash Faults (MAPPCF)

MAPF where an agent may stop moving indefinitely, but the other agents are still required to reach their goals via alternate pre-planned paths.
Related problems: k-robust (⋄delay type, ⋄alternate paths); OTIMAPP (-alternate paths, ⋄unbounded uncertainty)

In MAPPCF, some agents may break and stop moving forever (Okumura and Tixeuil 2023). The other agents should still be able to reach their goals without replanning. This problem is similar to OTIMAPP, but lifts the assumption that the agents eventually start moving. Therefore, simply preventing deadlocks is insufficient. The solution is a set of paths for each agent and a set of the so-called *transition rules*, which define when the agent changes its active path. This makes the problem similar to the contingency planning approach to k-robust MAPF (Section 6.2).

Agents can switch paths using the so-called *failure detector*, which can tell an agent whether a neighboring agent has crashed or not. Two types of failure detectors were studied: anonymous, which can identify only whether the agent has crashed or not, and named, which can also identify which agent it is. The named failure detector is important for sequential execution in which the agents move one after another (not at the same time) where the anonymous detector can lead to a failure. This is because the agent may have two options, but which option is correct depends on the other agents. Knowing which of the agents crashed can help to choose. The benefit of the named detector paired with synchronous execution is not yet determined, but it is not worse than that of the anonymous detector, because the named detector can act the same by simply ignoring the information about which agent crashed. MAPPCF was solved using a suboptimal method that minimizes flowtime and traveled distance (equal to flowtime without waiting actions) (Okumura and Tixeuil 2023).

[10] OTIMAPP solvers & visualizer: https://github.com/Kei18/otimapp/
[11] Anonymous OTIMAPP solver: https://github.com/Kei18/tswap

6.7 Robust Execution

Given a MAPF plan, the goal is to execute it safely even if the agents become arbitrarily delayed.
Related problems: OTIMAPP (⋄arbitrary delay); ACID (single instance in time)

The problem of robust execution is about safely executing classical (potentially non-robust) MAPF plans even if the fleet is not perfectly synchronized (Hönig, T. K. Kumar, et al. 2016). Although it is not a typical MAPF flavor, robust execution is complementary to MAPF. Perfect fleet synchronization is hard to enforce in practice for many reasons, such as imprecise localization, control, kinodynamic differences between the robots and external factors such as unexpected obstacles (fallen items on the ground) or human interference. These factors can cause some robots to become disproportionately delayed compared to the rest of the fleet, introducing a collision risk. Robust execution methods take MAPF plans as input and ensure that commands are given to agents only when it is safe to execute them.

In addition to safety, there are also other important aspects of the execution, such as its effectiveness regarding communication. With kinematically constrained agents, executing some plans may require frequent communication between the agents to ensure that they execute their actions safely, since they may tend to deviate from the plan. By postprocessing the plan to create an execution schedule, which takes into account the kinematic constraints, we can reduce the necessary amount of communication between the agents (Hönig, T. K. Kumar, et al. 2016). If there is a possibility that the original plan may become invalid and it is necessary to replan, *persistence* (Hönig, Kiesel, et al. 2019) becomes important. An execution is *persistent* if there are no unnecessary wait times, such as agents that could follow their old plans but are kept waiting because the replanning solver is still looking for a new plan.

Recently, there has been much interest in optimizing execution duration by rescheduling the order in which agents enter crossroads to minimize the impact of delays (Berndt, Duijkeren, et al. 2024; Berndt, Van Duijkeren, et al. 2020; Jiang, Lin, et al. 2025; Zahrádka, Kubišta, et al. 2023). Many of these methods expand on the Action Dependency Graph (Hönig, Kiesel, et al. 2019), which is itself a robust execution method. A Receding Horizon approach for online optimization of the Action Dependency Graph is presented in (Berndt, Duijkeren, et al. 2024), which optimizes the order of agents at crossroads for a fixed number of future timesteps. An improvement with better scaling capability was presented later (Jiang, Lin, et al. 2025). A similar approach was presented in (Su et al. 2024), which optimizes the execution time by giving priority at the crossroads to the agents who arrive there first, while maintaining safety of the execution. An algorithm based on A* for the same purpose was presented in (Feng et al. 2024). In (Zahrádka, Kubišta, et al. 2023), the execution time is optimized by formulating the execution of MAPF plans as a Job Shop Scheduling Problem (H. Xiong et al. 2022), which is solved with a metaheuristic. Recently, an ADG-based method was presented to predict future delays and their impact on execution time, allowing for pre-emptive resolution (Zahrádka, Mužíková, et al. 2025).

A problem dual to robust execution was also addressed: deciding on the agents' actions during the execution with an online planner, instead of executing an existing plan (Okumura, Tamura, et al. 2021). This leads to the problem of Time-Independent Planning, in which the goal is to decide which actions the agents should take while ensuring that they do not lead to collisions or deadlocks. An offline version of this planning problem is OTIMAPP (Section 6.5). Another related work deals with a situation where a robust execution method was not used, the plan failed as a result, and the goal is to compute how much each agent's delay contributed to the failure (Natan et al. 2023).

7 MAPF in Special Environments

In traditional MAPF, the spatio-temporal environment where the agents operate is discrete. The agents move on a graph; they can perform exactly one action per time step, each action takes exactly one time step, and they all move at once.

Some of these assumptions are simple and align with real-world robotic applications. For example, discretizing space is a straightforward task and is a common practice for robotic applications. However, the discretization step (which defines the width of the cells in the grid map) is usually different from the size of the robots. In such a case, an agent would not neatly fill exactly one cell of the map, and objects of interest may also not align perfectly with the centers of the cells. This may cause issues because an agent visiting the location of interest may partially occupy some of the neighboring cells as well. If we compensate for the misalignment by selecting a larger discretization step, we risk losing resolution in narrow areas, which might make them untraversable.

Other assumptions, such as the uniform travel times across all edges, are simply difficult to enforce in practice. The edges in the graph may represent different distances in the real world or some areas may require slower traversal. Having different durations of actions makes it difficult or even impossible to fit them neatly into discrete timesteps.

For such cases, there are problem formulations that assume a different *environment* for the agents, such as an environment with continuous time or space.

Table 4. Different features of MAPF flavors with special environments.

Problem	Section	Generalized Space	Generalized Time	Notes	Method
Any-angle	7.1	×	×	Agents can move to any vertex within line of sight	CO
N-MAPD	7.2	✓	×	Carrying an item disables some edges	O
$MAPF_R$	7.3	✓	✓		RCO
CE-MAPF	7.4	✓	✓		CO

Table 4 presents the collected problems and their defining characteristics. The first class of problems has generalized space, meaning that the agents operate in an environment not represented by a standard four-connected graph, where each vertex has up to four neighbors (one for each cardinal direction) or that the graph has some specific properties not present in classical MAPF. The second class is problems with generalized time, which lift the assumption that time is discretized into timesteps. All of the presented problems generalize collisions as well, because simply checking whether two agents occupy the same vertex at the same timestep or exchanged places is insufficient when actions do not take exactly one timestep, the agent may occupy multiple grid cells at the same time when moving over them, or when the environment is not a graph at all.

7.1 Any-angle MAPF

MAPF where agents can move into any cell connected by a straight line that does not intersect an obstacle.
Related problems: $MAPF_R$ *(+arbitrary edge travel times, +general graph environment);* CE-MAPF *(+continuous space);* MAPFKC *(+motion model, ⋄turning);* $MAPF_T$ *(-gen. collisions, -kinematics, ⋄turning);* LA-MAPF *(-kinematics, +arbitrary agent shape)*

The environment in Any-angle MAPF is a grid where the neighbors of each cell are all cells that can be connected via a straight line segment (Yakovlev and Andreychuk 2017) compared to four cardinal neighbors (and potentially four more diagonal neighbors) in classical MAPF. The lines connect the centers of the cells and they may not cross an obstacle. Thus, agents can move to any vertex within line of sight. Since the edges in the resulting graph may represent different distances, Any-angle MAPF involves dealing with the agent's *action time*.

Each agent travels with the same speed, but due to the different distances, different actions may take different amounts of time, such as more than one time step. Furthermore, an additional issue is collision detection: Since the agent is traveling at an angle, it might occupy portions of different cells at the same time. It is therefore necessary to represent collisions geometrically, using the size of each agent. Any-angle MAPF was solved using optimal and bounded-suboptimal modifications of CBS and ECBS for flowtime (Yakovlev and Andreychuk 2017).

Any-angle MAPF was also combined with Offline MAPD (Section 4.13). The problem was solved with a decoupled approach that first assigns tasks using Hungarian algorithm and then plans paths with a prioritized Any-Angle SIPP (Yakovlev, Andreychuk, Rybecký, et al. 2020). The criteria used were makespan and flowtime.

7.2 MAPD under Environmental Constraints (N-MAPD)

MAPF with multigoal TA where each goal has pickup and delivery locations, different dimensions and a subset of edges cannot be traversed while carrying items that are too large.
Related problems: *DD-MAPD (-gen. environment, ⋄traversability); MAPD (-gen. environment, +lifelong)*

N-MAPD is a variation of MAPD (Section 5.3) where the traversability of the environment is constrained (Yamauchi et al. 2022). It is motivated by applications where autonomous robots with carrying capabilities transport items through obstacle-dense environments. The obstacles limit the movement of the robots under some circumstances. If a robot is carrying an item, it may not fit in narrow corridors where it would be able to go otherwise. This problem is similar to DD-MAPD (Section 4.13), but it has significant differences. Carrying an item in both problems means that the agent cannot perform some moves. In DD-MAPD, the agent cannot drive under shelves while carrying one, and in N-MAPD, it cannot move into some parts of the map. However, in DD-MAPD, the positions of the shelves may change, while in N-MAPD, the narrow corridors remain in the same place. Hence, N-MAPD and DD-MAPD do not belong to the same category of flavors. Additionally, in N-MAPD, the items can have varying sizes, and thus carrying some items restricts the agent more than other items. The problem was solved using a suboptimal multilevel method with makespan and operational time – the average time to complete a task – as criteria (Yamauchi et al. 2022).

7.3 MAPF with Continuous Time (MAPF_R)

MAPF with arbitrary edge travel times.
Related problems: *Any-angle (-arbitrary edge travel times, -general graph environment); CE-MAPF (+continuous space); MAPF-TU (+uncertainty, ⋄edge costs); MAPFKC (+motion model, ⋄non-unit action duration); UAVs (-gen. time, ⋄collisions)*

One of the strong assumptions in MAPF is the discrete time, due to which agents can perform one action per time step. Relaxation of this assumption yields MAPF_R (Walker et al. 2018). In MAPF_R, the move and wait actions have a non-uniform duration represented by non-uniform real-valued edge weights. The motivation is to be able to plan on any connected graph where edges may potentially have different weights (lengths), representing, for example, the distances between arbitrarily defined vertices. Additionally, the agent's shape is taken into account, and the edges of the graph representing the environment may intersect each other. That means that two agents moving between unrelated vertices may collide with each other while on their way, and it is not sufficient to check for vertex and edge collisions. While this may resemble agents moving in continuous *space*, there is still the assumption that the agents move on a graph and the agents always move between its vertices. This distinguishes MAPF_R from the Continuous Space MAPF described in Section 7.4.

The problem was solved in (Walker et al. 2018) using increasing cost tree search with flowtime criterion, but with the assumption that the wait actions have a fixed duration: the agents cannot wait for an arbitrarily long time, they can only chain together fixed-length wait actions. The method can also be adapted to makespan. A

method capable of using waiting actions of variable length based on CBS was presented in (Andreychuk, Yakovlev, Boyarski, et al. 2021; Andreychuk, Yakovlev, Surynek, et al. 2022), again for both common criteria. Furthermore, the problem can be solved for both criteria with a reduction-based approach (Surynek 2021a).

There is also a continuous-time version of k-robust MAPF (Section 6.2) called T-robust MAPF (Tan et al. 2024). The solutions to T-robust MAPF must remain safe even if a number of agents are delayed by a length of T. The problem was solved with a modification of CBS for makespan.

7.4 MAPF with Continuous Space/Smooth Continuous MAPF (CE-MAPF)

MAPF where the agents move in a continuous environment in contrast to a graph.
Related problems: Any-angle (-continuous space); $MAPF_R$ (-continuous space)

Further relaxation of the discrete environment assumption is present in MAPF in Continuous Spaces (Liang et al. 2024), also called CE-MAPF (Janovská and Surynek 2024). In this problem formulation, the agents do not move on a graph, but on samples of the continuous obstacle-free space. The samples are generated by the solvers, and an agent's path is a sequence of curves between the samples. Collisions are geometrical, which means that the distance between any agent and any obstacle (static, or dynamic, such as another agent) must be larger than some value, such as the diameter of the agent. The agents are also kinematically constrained. In (Kasaura et al. 2023), a variation of Online MAPF (Section 5.1) with continuous space was studied in which new agents periodically appear.

In (Liang et al. 2024), the problem was solved using a diffusion model that minimized flowtime. Another approach that uses a CBS modification that also minimizes flowtime is presented in (Janovská and Surynek 2024).

8 MAPF with Special Agent Models

Another way to modify MAPF is to change the behavior of the agents. Typically, MAPF models agents as an entity that occupies one vertex at a time and can move to any single neighboring vertex in each step. In contrast to MAPF with action uncertainty, which only considers the probability of failing an action, or getting delayed while executing it, this category of problems deals with the agent's behavior itself. For example, the agent may be subject to movement constraints, such as not being able to rotate in place. Another example can be that the agent is larger than one vertex, and occupies multiple vertices at all times. This has some similarities to k-robust MAPF (Section 6.2), where an agent occupies all vertices it visited in the last k timesteps, but here, the agent may not fit into some corridors at all.

Table 5. Different features of MAPF flavors with special agent models.

Problem	Section	Kinematics	Motion Model	Generalized Collisions	Dimensions	Notes	Method
$MAPF_T$	8.1	×	Nonholonomic	×	2		C
LA-MAPF	8.2	×	Arbitrary	✓	2,3		C
UAVs	8.3	×	Classical	✓	3		RC
Trains	8.4	×	Varies	✓	2	Self-collisions	RCO
MAPFKC	8.5	✓	Nonholonomic	✓	2	Steering models	CO

Table 5 summarizes the existing agent-based MAPF flavors. The problems have multiple characteristics: some consider kinematic constraints of the agents. The problems often use different motion models: we recognize the Classical model, in which an agent can move over any edge connected to its vertex, and the Nonholonomic model, in which the agent's orientation matters. Some nonholonomic agents have differential drive steering, while others may use more complex models, such as the Ackermann steering model. Some problems feature a

combination of different models, and one problem even allows arbitrary definition of movement rules. Almost all the presented problems require *generalized collisions*, and in most of the problems, the agents move in a 2D world.

8.1 MAPF with Turn Actions (MAPF$_T$)

MAPF where the agents can drive only forward and turning is a separate action.
Related problems: *Any-angle (+gen. collisions, +kinematics, ⋄turning);*
MAPFKC (+gen. collisions, +kinematics, ⋄turning)

In MAPF with Turn Actions (MAPF$_T$) (Y. Zhang, D. Harabor, et al. 2023), also called MAPF with Rotations (MAPF-R) (Jiang, Y. Zhang, et al. 2024), the agents have a different set of actions. Instead of being able to move up, down, left, or right, the agent's options are defined by its orientation. In each timestep, the agent can only move forward or rotate 90 degrees to the left or right, or stay waiting in its current vertex. Unlike other variations dealing with the agent's orientation and therefore turning or steering, such as Any-angle MAPF (Section 7.1) or MAPFKC (Section 8.5), MAPF$_T$follows classical MAPF rules: every action takes exactly one timestep and the agents move only between neighboring vertices. Therefore, classical MAPF conflict rules apply because it is not necessary to use a generalized collision model.

In (Y. Zhang, D. Harabor, et al. 2023), an optimal CBS-based solver for flowtime was presented that is more effective for solving MAPF$_T$. A lifelong (Section 5.2) variant of MAPF$_T$ exists (Y. Zhang, D. Harabor, et al. 2023), which is used as the problem formulation for the League of Robot Runners competition[12] (Chan et al. 2024).

8.2 MAPF with Large Agents (LA-MAPF)

MAPF where agents can have any geometric shape and it can change.
Related problems: *MTPF (⋄collisions); k-robust (+uncertainty, ⋄collisions); Any-angle (-arbitrary agent shape, +kinematics)*

LA-MAPF is a problem in which each agent can have an arbitrary fixed geometric shape (J. Li, Surynek, et al. 2019). It is assumed that the agents do not rotate, and thus, their orientation remains constant. The problem formulation also allows agents representing 3D objects operating on roadmaps in 3D space. The main complication of this variation is collision detection, as the agents do not occupy only one vertex. This is similar to Any-angle MAPF (Section 7.1), where agents can occupy multiple cells while moving. The differences are that LA-MAPF does not consider kinematic constraints, and Any-angle MAPF assumes that the agents are circular. The authors solved the problem with a modification of CBS for the flowtime criterion.

A step further was presented in (Atzmon, Zax, et al. 2020), which formulated a MAPF variation for general heterogeneous agents. It allows defining a specific set of actions for each agent. Each action prescribes a transition from one state to another, where state is a set of occupied vertices. This can be any transition, such as movement, rotation, or even changing shapes. A conflict is then defined as two agents occupying a common vertex or two agents swapping locations. In (Atzmon, Zax, et al. 2020) it is shown that LA-MAPF with general heterogeneous agents can be used to express both LA-MAPF using the vertices occupied by the projection of the agent's geometric shape and k-robust MAPF (Section 6.2) by using a state of size k and a transition function that pops the $k + 1$th element while adding the vertex where the agent is at the end of the current timestep. The extended problem was also solved using a modification of CBS with flowtime as criterion.

[12] League of Robot Runners: https://www.leagueofrobotrunners.org/

8.3 MAPF for Unmanned Aerial Vehicles

MAPF where the agent is a UAV, often in a 3-dimensional space.
Related problems: MAPFKC (+kinematics, ⋄collisions); $MAPF_R$ (+gen. time, ⋄collisions)

The problem of UAV traffic control was formulated as a variation of MAPF in (Ho, Goncalves, et al. 2019). Each agent can have different speeds and sizes, and collisions are defined as a violation of minimum separation between two agents. Another difference from existing problems is that the agents can share starts and goals, but their start times differ. To avoid collisions, agents leave their starting location (take off) at different times. After reaching the goal location, the agents have to return to their starting locations. Furthermore, the agents operate in a 3D space consisting of cubic voxels, where each voxel is one vertex of a graph. Each agent can occupy one or more voxels, or even less than one voxel. The problem was solved optimally and bounded-suboptimally with a modification of CBS and ECBS for the flowtime criterion. A distributed solver for a similar problem with multiple path-planning entities, each managing UAVs in its sector, was also presented (Ho, Geraldes, et al. 2022). Another approach for UAVs was used in (Hönig, Preiss, et al. 2018), which studied the coordination of a UAV swarm in a smaller area. The smaller area requires dealing with characteristics inherent to UAVs, such as downwash. The possible maneuvers of UAVs were encoded in a roadmap. The roadmap had labeled edges to encode all possible other vertices or edges that can result in a collision if occupied. Then, CBS and a method based on reduction to Network Flow were used to find makespan-optimal paths on the roadmap. Afterwards, the paths were converted into smooth trajectories.

8.4 MAPF with Trains and Convoys

MAPF where the agents have long bodies, possibly prone to self-collisions.
Related problems: k-robust (+uncertainty, ⋄collisions); LA-MAPF (⋄collisions); Online MAPF (-special agent model, ⋄varying agents)

MAPF was also formulated for trains of fixed (J. Li, Z. Chen, Zheng, et al. 2021) and variable (Atzmon, Diei, et al. 2019; Švancara and Barták 2023) lengths. The problem solved in (J. Li, Z. Chen, Zheng, et al. 2021) was the subject of the FLATLAND competition[13], in which the goal was to maximize the number of trains that reached their depots within a deadline, similarly to MAPF-DL (Section 4.1). Each agent occupies one vertex and its movement is defined by the rails. The agent can move forward, wait, or turn left or right on a junction, depending on the junction type. Being a competition-motivated problem, it also involved other issues, such as agents appearing and disappearing much like in Online MAPF or DMAPF (Section 5.1) and execution (Section 6.7). The problem was solved with a specialized suboptimal method (J. Li, Z. Chen, Zheng, et al. 2021). Two problem formulations that more closely reflect the general behavior of trains are presented in (Švancara and Barták 2023). In the first, each train consists of a group of agents that must behave as the train's wagons. That means they must i) remain connected at all times and ii) each agent except the first may occupy only the vertex that the agent in front of them did in the last time step, or remain in place if the train is waiting. In the second case, the problem is formulated using a highly abstracted *scheduling model*. The problem was solved optimally for makespan.

Another formulation of the problem is presented in (Atzmon, Diei, et al. 2019), which formulates the Multi-Train Path Finding (MTPF) problem. The difference between MTPF and the problem presented in (Švancara and Barták 2023) is that the start and goal locations model train depots from which the train leaves and at the end drives into, occupying only a single vertex afterward. This problem was solved optimally with a modification of CBS for flowtime.

[13]FLATLAND Competition: https://www.aicrowd.com/challenges/flatland

Train-related formulations are very close to k-robust MAPF (Section 6.2). The main difference is that, in k-robust MAPF, a waiting agent progressively occupies fewer and fewer vertices. This is because it occupies the vertices of its last k actions. After waiting for k timesteps, it occupies only one vertex. However, a train of length k would still occupy k vertices, since its length does not change by waiting.

A closely related problem is MAPF for convoys (Thomas et al. 2015). Convoys move over edges and can evade each other in parking locations (vertices). One convoy can occupy multiple edges at the same time, and multiple convoys may travel along the same edge if they maintain a defined minimum distance between each other. The problem takes into account other constraints, such as maximum speed and the requirement that the convoy must occasionally spend some time in parking locations. The similarity to the train model is that once a convoy enters an edge, it can only travel forward to its sink, similar to self-collisions. The convoy problem was solved with an extension of CBS for the flowtime criterion.

Another related problem is Non-Crossing Anonymous MAPF (NC-AMAPF), in which the agents are tethered with taut cables to their starting positions (Peng et al. 2023). The tethers may not cross and, therefore, the agents' paths may not cross as well: they occupy all vertices they move across, similarly to an infinite-length train. While the problem is set in a continuous environment, it is assumed that the agents move on a visibility graph of the environment. This is because the cables are kept taut, and therefore the visibility graph represents all possible positions of the cables. The goals are fully anonymous, similar to AMAPF (Section 4.6). The problem was solved suboptimally with a metaheuristic approach and optimally with reduction to Constraint Programming for makespan (Peng et al. 2023).

8.5 MAPF with Kinematic or Dynamic Constraints (MAPFKC)

MAPF with nonholonomic agents and velocity or acceleration limits.
Related problems: *$MAPF_T$ (-gen. collisions, -kinematics, ⋄turning); Any-angle (-motion model, ⋄turning); $MAPF_R$ (-motion model, ⋄non-unit action duration); UAVs (-kinematics, ⋄collisions)*

In MAPF with Kinematic Constraints (MAPFKC) (Yakovlev, Andreychuk, and Vorobyev 2019), the movement of the agents is subject to velocity or acceleration limits, which can be different for each agent. Furthermore, the headings of the agents matter (similar to $MAPF_T$ in Section 8.1) and so does their shape. This problem is very close to $MAPF_R$ (Section 7.3) and Any-angle MAPF (Section 7.1), since they also often use agents with limited speed. Kinematics are used in these problems because the travel times over edges are nonunit, and to detect collisions, we need to know exactly when and where the agents collide. However, both Any-angle MAPF and $MAPF_R$ assume that the travel times are defined by the edge length and not by the agent's kinematics.

Initially, dynamic constraints were not considered and agents always moved at constant speed. The problem was solved with a modification of Prioritized SIPP for both flowtime and makespan (Yakovlev, Andreychuk, and Vorobyev 2019). The method is available online[14]. An Any-angle CBS-based method to find a flowtime-optimal solution was used in (Andreychuk 2020). In (Ali and Yakovlev 2021), Prioritized SIPP with flowtime as criterion was used to solve MAPFKC with dynamic constraints, adding acceleration limits. In (Ali and Yakovlev 2023), it is explained that the SIPP used as a local planner is incomplete in the presence of dynamic constraints, and a modification that solves the issue is presented. This modification is available online[15].

In (Wen et al. 2022), an Ackermann steering model was used to formulate MAPF for Car-like Robots (CL-MAPF) moving in continuous space. In contrast to the commonly used differential drive, the Ackermann steering model has a non-zero minimum turning radius, making maneuvers more complex, as the agents cannot rotate in place. This also means that the start and goal states have a defined heading. CL-MAPF was solved with a modification

[14]MAPF with Kinematic Constraints Solver: https://github.com/PathPlanning/AA-SIPP-m
[15]SIPP-IP: https://github.com/PathPlanning/SIPP-IP

of CBS with makespan and average path length as a criterion. The CBS used motion primitives at its low level to deal with continuous space. Later, the algorithm was also extended to bounded-suboptimal and suboptimal versions with support for heterogeneous agents, where each agent can have a different steering model (Bai et al. 2025).

By combining MAPFKC with continuous time and space, we could obtain the Multi-Robot and Multi-Agent Motion Planning problems (Yan and J. Li 2025). However, since these problems originated in the robotics community, we consider these problems too distant from MAPF to include in this survey, even when there is some overlap in the problems. There are also many different solution methods, such as various sampling-based approaches. Their summary would be too extensive to include here and would be a better fit for a standalone survey.

9 Unique Problems

This Section contains flavors with their defining features unique enough that they fall outside the previously defined categories. These, for example, include problems other than collision-free path planning or MAPF with unique criteria and multiple objectives. Since these problems do not commonly share defining characteristics, we only provide a brief summary of each of them in Table 6.

Table 6. Summary of unique problems.

Problem	Section	Description	Method
PERR	9.1	Swap collisions allowed	RC
ACPF	9.2	Two teams, one tries to reach goals, other tries to prevent it	O
CF-MAM	9.3	Finding a meeting point for a team of agents	RC
MOMAPF	9.4	Multiple optimization criteria, search for Pareto optimality	C
MAiF	9.5	Agents maintain a formation while moving	CO
Social MAPF	9.6	Agents have private self-centered interests	CO
ACID	9.7	Fix MAPF solutions with collisions by adding delays	RC
CC-PP	9.8	Agents must remain within maximum distance of each other	RO
Analysis	9.9	Analyze solutions and instances to gain information	O

9.1 Package-Exchange Robot Routing (PERR)

MAPF where swap conflicts are allowed.
Related problems: N/A

PERR is a problem in which there are packages located at source locations, and they need to be carried to the destination locations (Ma, Tovey, et al. 2016). It is assumed that each package is carried by exactly one robot and that each robot carries one package. Therefore, package transport is modeled as traditional MAPF, with packages being modeled as agents. There is a significant distinction in the form of the *exchange* operation, where two adjacent robots swap packages. This is equivalent to allowing for the swap conflict in classic MAPF, and therefore any MAPF solver can be used. The problem is motivated by a package delivery scenario where robots can exchange packages, or by ride-sharing with passenger transfers. By expressing the carried object as the agent rather than the carrying robot, it becomes easier to solve the problem: unlike MAPF, all PERR instances are solvable (Ma, Tovey, et al. 2016), provided that the underlying graph is connected. PERR has been solved for flowtime and makespan with a Network Flow and CBS-based method.

9.2 Adversarial Cooperative Path Finding (ACPF)

MAPF where one team of agents tries to complete the goals while the other team tries to prevent it.
Related problems: N/A

In ACPF, there are multiple teams of agents (Ivanová and Surynek 2013). One team is the team of interest, and the rest are adversarial teams. Each team aims to be the first to reach all of its goals while preventing other teams from reaching theirs by, for example, blocking some of the other team's agents or goal locations. The teams take turns moving, all agents being able to make one action per turn. The goal is to decide whether there is a winning strategy that would ensure that the team of interest is the first to complete its goals. An asymmetric version of the problem is presented in (Ivanová and Surynek 2021), in which only the first team aims to reach its goals, whereas the second team attempts to block either the first team's agents or its goal locations. The first team wins if it reaches all its goals and the second team wins if it successfully prevents the first team from reaching one or more of the goals.

The symmetric version of the problem is PSPACE-hard for two teams (Ivanová and Surynek 2014) and the asymmetric version is EXPTIME-complete (Ivanová and Surynek 2021). The opposite problem is area protection (Ivanová, Surynek, and Hirayama 2018), where there is a team that wants to *capture* a set of locations (goals) and there is a *defending team* that tries to prevent it. Multiple objective functions are proposed, such as maximizing the number of locations not captured with or without a time limit, maximizing the energy expenditure of the capturing team, or minimizing the time spent at the captured targets. This problem is also PSPACE-hard (Ivanová, Surynek, and Hirayama 2018). ACPF was solved using different methods from the field of game theory and a modification of a suboptimal search-based method for MAPF (Ivanová and Surynek 2013).

9.3 Conflict-Free Multi-Agent Meeting Problem (CF-MAM)

MAPF where all agents have the same goal – a meeting point.
Related problems: Co-MAPF (subproblem of)

The goal of CF-MAM is to find the best *meeting point* for a group of agents (Atzmon, Felner, et al. 2023). There are two variants: conflict-tolerant, in which more than one agent may occupy a vertex at one time, and conflict-free (CF-MAM), in which they can occupy only the meeting location simultaneously. CF-MAM, therefore, involves finding a set of collision-free paths for the agents to the location (solve MAPF). This problem is distantly related to Co-MAPF (Section 4.8), in which pairs of cooperating agents have to meet each other at some vertex to exchange an item that one picked up and the second must deliver. The problem was solved with a method using reduction to Network Flow and a modification of CBS for flowtime and makespan.

9.4 Multi-Objective MAPF (MOMAPF)

MAPF with multiple optimization criteria where the goal is to find a Pareto-optimal set of solutions.
Related problems: MAiF (+positioning requirements, ⋄multiple objectives); Social MAPF (-Pareto optimality, +hidden criteria)

A variation of MAPF that focuses on minimizing multiple criteria at the same time is MOMAPF (Z. Ren, Rathinam, et al. 2021b). In contrast to classic MAPF, where each edge (action) has a scalar cost, the edges in MOMAPF have *cost vectors*, with one value for each criterion. Since there may be a trade-off between optimizing different criteria, it is difficult to define what constitutes an optimal solution. The goal is therefore to find a so-called Pareto-optimal set of solutions. A Pareto-optimal solution is such that it is not possible to improve any criterion without negatively affecting the other criteria. By obtaining this set, it can be said that no matter which criterion is preferred, it contains the optimal solution for such a preference.

The problem was solved using multi-objective modifications of CBS (Z. Ren, J. Li, et al. 2023; Z. Ren, Rathinam, et al. 2023a, 2021b,c). The instances are not available online, but are described in (Z. Ren, Rathinam, et al. 2023a) and the method is publicly available[16]. Alternatively, (Matsui 2025) focuses on a different criterion that expresses fairness and minimizes the worst-case solution for each agent.

A modification of MOMAPF is the Multi-Agent Teamwise Cooperative Path Finding problem (TCPF), in which there are multiple teams, each with its own objective (Z. Ren, Cai, et al. 2024; Z. Ren, Rathinam, et al. 2023b). The objectives can be, for example, flowtime or makespan, and each agent belongs to at least one team. TCPF was solved with an incomplete modification of classical CBS (Z. Ren, Rathinam, et al. 2023b) and later with an improved complete version (Z. Ren, Cai, et al. 2024).

9.5 Moving Agents in Formation (MAiF)

MAPF where the agents maintain a formation while moving to their goals.
Related problems: MOMAPF *(-positioning requirements, ⋄multiple objectives);* CC-PP *(-multiple objectives, +lifelong, ⋄positioning requirements)*

MAiF is an interesting problem closely related to MOMAPF (J. Li, Sun, et al. 2020). The goal is to minimize makespan while also having the agents maintain a desired formation while moving. The desired formation is defined by the positions of the goal locations. The agents can deviate from the formation, for example when there is an obstacle in the way, but it is penalized: adherence to the formation is a second optimization criterion, hence the resemblance to MOMAPF.

The problem was solved by a method that designates a leader and plans its path while minimizing the formation deviation (J. Li, Sun, et al. 2020). When some agents have to deviate from the formation due to an obstacle, a CBS-based method was used. The CBS-based method can also be used to solve MAiF on its own. The authors also show that finding the Pareto-optimal set is too time-consuming.

9.6 Social MAPF

MAPF where agents have private criteria in addition to a shared global optimization criterion.
Related problems: MOMAPF *(-hidden criteria, +Pareto optimality)*

Social MAPF removes the assumption that the fleet is cooperative by introducing *strategic* agents (Chandra et al. 2023). These agents have their own private incentives that are only known to them. It is inspired by human motivation: an agent representing a person rushing to the hospital has a higher incentive than someone who goes to the grocery store. The goal is to find collision-free paths that minimize flowtime, but at the same time maximize utility gain for the agents.

Although Social MAPF resembles a specific kind of TCPF where each team has only one agent, there is the important difference that the agents in Social MAPF may not be cooperative and agents do not know the incentives of other agents. This means that a fully central planner cannot be used to solve the problem, because it lacks the agents' private information. Instead, a combinatorial auction combined with CBS was used, in which the agents bid for their priority in conflict resolution according to their incentives (Chandra et al. 2023). The low-level planner is an Artificial Potential Field method.

[16]MOMAPF solver: https://github.com/rap-lab-org/public_cppmomapf

9.7 Avoiding Collision by Introducing Delays (ACID)

The input is a MAPF solution that contains collisions and the goal is to resolve them by adding waiting actions.
Related problems: Execution (ongoing collision resolution)

ACID (Kottinger et al. 2024; Nekvinda and Barták 2021; Švancara, Tignon, et al. 2023) is an APX-hard problem in which, during the execution of a MAPF solution, some agents have been delayed in such a way that continuing the execution would cause a collision. In other words, ACID starts with a MAPF solution that contains collisions. The goal is to resolve the collisions by only adding waiting actions to the agents' existing paths. The motivation is that reusing the original paths is faster than fully replanning paths for the agents. ACID is very close to the problem of robust execution, and solving ACID whenever an agent is delayed can be used as a robust execution method. However, as a standalone problem, ACID assumes that the agents will follow the new plan perfectly.

A method for an ACID-like problem that focuses on minimizing makespan was presented in (Nekvinda and Barták 2021), which redistributes already present wait actions in a plan to increase its robustness. An approach to solving ACID by constructing an *extended wait-graph* and checking whether a solution exists using Answer Set Programming also exists (Švancara, Tignon, et al. 2023). A CBS-based method on a special graph was used to optimally solve the problem for the flowtime criterion (Kottinger et al. 2024).

9.8 Communication-Constrained Multi-Goal Path Planning (CC-PP)

MAPF where the agents must remain within communication distance of at least one other member of the fleet.
Related problems: LMAPF (-positioning requirements); MAiF (-lifelong, +multiple objectives, ⋄positioning requirements)

CC-PP is an extension of LMAPF (Section 5.2) in which robots must stay within a maximum distance of each other (Herynek and Edelkamp 2024). This models a scenario where the fleet is collecting data and there is one robot capable of path planning and cataloguing the collected data. The robots can relay messages to each other, but all must remain within some maximum distance of another connected member of the fleet. The similarity to LMAPF is that the planning-capable robot assigns goals to members of the fleet, making it a time-extended assignment problem. Another similar problem is MAiF (Section 9.5), where the agents are required to maintain a formation while moving, and their deviation from the formation is an optimization variable. However, in CC-PP, there is no explicit requirement that the agents maintain a specific configuration as long as they remain within communication range. Moreover, in contrast to CC-PP, MAiF is not a time-extended assignment problem. Another similar problem is Multi-Agent Path Finding with Communication Constraints (MAPFCC) (Fioravantes et al. 2026), in which the agents are required to navigate to their goals while remaining *connected*. The agents have a maximum communication distance of d and they must move in a single group where no agent is more than d steps away from at least one other member of the group, forming a communication chain. However, unlike CC-PP, MAPFCC is not a time-extended assignment problem.

CC-PP was solved with a suboptimal method for makespan (Herynek and Edelkamp 2024). An offline variant of the problem was studied in (Bui et al. 2025), which was solved with a suboptimal prioritized approach minimizing traveled distance (flowtime without waiting actions). MAPFCC was solved for makespan by reduction into a Monadic Second Order logic formula (Fioravantes et al. 2026).

9.9 Analysis of MAPF Instances and Solutions

Problems that involve analyzing MAPF maps, instances and solutions, commonly to select the right solver.
Related problems: N/A

Some research focuses on *analysis* of MAPF instances and solutions, rather than solving MAPF. The most common type of methods for analysis of MAPF focuses on *algorithm selection*. Many methods perform exceedingly well in their specific niche, but can be outperformed outside of it. Furthermore, most methods have parameters that can be adjusted to better fit the task at hand. Perhaps the most notable example is the suboptimality factor of bounded-suboptimal solvers, which, if set too low, can prevent finding a solution, but set too high can produce worse results than achievable. Therefore, selecting the right method for the task based on the required instance, or estimating the difficulty of the instance, can be crucial for applications.

Two approaches were used: Convolutional Neural Networks (Alkazzi, Rizk, et al. 2022; Kaduri et al. 2020; J. Ren et al. 2021; Sigurdson, Bulitko, Koenig, et al. 2019) and XGBoost, a tree-based learning algorithm (Ewing et al. 2022; Kaduri et al. 2020). For the CNN-based approach, it was beneficial to embed information about the shortest path for each agent into the instance (J. Ren et al. 2021). An improved version of the algorithm (Alkazzi, Rizk, et al. 2022) and the training data are available online[17]. For the tree-based learning approaches, post-processing the instances by computing the *betweenness centrality*, which expresses the probability of each vertex belonging to the shortest path between any two vertices, has proven useful (Ewing et al. 2022). The betweenness centrality can be used both to analyze an instance's difficulty and as a feature for algorithm selection. These methods usually consider flowtime, success rate, or solution speed as a criterion to compare solution quality.

Another subject of interest was *vulnerability* of solutions (Yoeli et al. 2021). It is motivated by a scenario where an agent is hijacked by a malicious actor. The malicious actor causes the agent to perform a limited number of specific actions that were not part of its original plan. After each action, a central controller generates a new, collision-free plan. The goal of the problem is to find the *maximum damage* (increase in makespan) that malicious actions can have on execution. It is formulated as a search problem and is solved using A* with a specialized heuristic.

10 Conclusion

In this work, we collected existing derivations of MAPF problems – the different *flavors* of MAPF – and separated them into categories. For each flavor, we identified its defining features, summarized them, and presented current state-of-the-art solutions for the problems. The resulting paper is aimed at three main categories of readers:

(1) New researchers interested in orienting themselves in the area of existing flavors of MAPF and their solution approaches.
(2) Experienced researchers that are interested in updating their knowledge in the area of MAPF flavors and finding new problems to solve.
(3) Readers who have a problem they are trying to solve and are searching whether someone solved it already, or whether someone was studying related problems, to see if there is an existing solution they could use.

All three types of readers can benefit from perusing the presented problems that are relevant to their fields of interest. As a concluding remark, we also present some statistics on the solution approaches to MAPF derivations. In total, we presented 48 different problems. The most common solution approach was using a variation of CBS, being used to solve 32 different flavors. Reduction to other problems, such as Network Flow, SAT, or Answer Set Programming, was used in 18 problems. Other methods that cannot be classified as CBS-based or reduction-based were also very common, used for 30 problems in total. These include prioritized planning, metaheuristics, and bespoke solvers designed specifically for the problem.

Regarding optimization criteria, flowtime was slightly more common, being used in 33 problems. Makespan was also used frequently, appearing in 27 flavors. Clearly, many problems were solved for both of these criteria. This is because flowtime-optimizing methods are also often capable of optimizing makespan and vice versa.

[17] MAPFASTER Algorithm selector: https://github.com/jeanmarcalkazzi/mapfaster

Finally, other optimization criteria, such as throughput or binary criteria that measure success, were used in 25 problems.

Acknowledgments

This work was supported by the Czech Science Foundation Grant No. 23-05104S, the Grant Agency of the Czech Technical University in Prague, Grant number SGS23/180/OHK3/3T/13, and by Charles University project UNCE 24/SCI/008.

References

R. A. Achá, R. López, S. Hagedorn, and J. A. Baier. Sept. 2022. "Multi-Agent Path Finding: A New Boolean Encoding." en. *Journal of Artificial Intelligence Research*, 75, (Sept. 2022), 323–350. doi:10.1613/jair.1.13818.

Z. A. Ali and K. Yakovlev. Mar. 2024. "Improved Anonymous Multi-Agent Path Finding Algorithm." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38, 16, (Mar. 2024), 17291–17298. Number: 16. doi:10.1609/aaai.v38i16.29676.

Z. A. Ali and K. Yakovlev. 2021. "Prioritized SIPP for Multi-agent Path Finding with Kinematic Constraints." en. In: *Interactive Collaborative Robotics* (Lecture Notes in Computer Science). Ed. by A. Ronzhin, G. Rigoll, and R. Meshcheryakov. Springer International Publishing, Cham, 1–13. ISBN: 978-3-030-87725-5. doi:10.1007/978-3-030-87725-5_1.

Z. A. Ali and K. Yakovlev. June 2023. "Safe Interval Path Planning with Kinodynamic Constraints." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 10, (June 2023), 12330–12337. Number: 10. doi:10.1609/aaai.v37i10.26453.

J.-M. Alkazzi and K. Okumura. 2024. "A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding." *IEEE Access*, 12, 57390–57409. Conference Name: IEEE Access. doi:10.1109/ACCESS.2024.3392305.

J.-M. Alkazzi, A. Rizk, M. Salomon, and A. Makhoul. Oct. 2022. "MAPFASTER: A Faster and Simpler take on Multi-Agent Path Finding Algorithm Selection." In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. (Oct. 2022), 10088–10093. doi:10.1109/IROS47612.2022.9981981.

A. Andreychuk. 2020. "Multi-agent Path Finding with Kinematic Constraints via Conflict Based Search." en. In: *Artificial Intelligence*. Ed. by S. O. Kuznetsov, A. I. Panov, and K. S. Yakovlev. Springer International Publishing, Cham, 29–45. ISBN: 978-3-030-59535-7. doi:10.1007/978-3-030-59535-7_3.

A. Andreychuk, K. Yakovlev, E. Boyarski, and R. Stern. May 2021. "Improving Continuous-time Conflict Based Search." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 13, (May 2021), 11220–11227. Number: 13. doi:10.1609/aaai.v35i13.17338.

A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern. Apr. 2022. "Multi-agent pathfinding with continuous time." en. *Artificial Intelligence*, 305, (Apr. 2022), 103662. doi:10.1016/j.artint.2022.103662.

B. Atiq, V. Patoglu, and E. Erdem. Sept. 2020. "Dynamic Multi-Agent Path Finding based on Conflict Resolution using Answer Set Programming." *Electronic Proceedings in Theoretical Computer Science*, 325, (Sept. 2020), 223–229. arXiv:2009.10249 [cs]. doi:10.4204/EPTCS.325.27.

D. Atzmon, A. Diei, and D. Rave. 2019. "Multi-Train Path Finding." en. *Proceedings of the International Symposium on Combinatorial Search*, 10, 1, 125–129. Number: 1. doi:10.1609/socs.v10i1.18515.

D. Atzmon, A. Felner, J. Li, S. Shperberg, N. Sturtevant, and S. Koenig. Sept. 2023. "Conflict-tolerant and conflict-free multi-agent meeting." *Artificial Intelligence*, 322, (Sept. 2023), 103950. doi:10.1016/j.artint.2023.103950.

D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig. June 2020. "Probabilistic Robust Multi-Agent Path Finding." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30, (June 2020), 29–37. doi:10.1609/icaps.v30i1.6642.

D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Bartak, and N.-F. Zhou. 2018. "Robust Multi-Agent Path Finding." en. *Proceedings of the International Symposium on Combinatorial Search*, 9, 1, 2–9. Number: 1. doi:10.1609/socs.v9i1.18445.

D. Atzmon, Y. Zax, E. Kivity, L. Avitan, J. Morag, and A. Felner. 2020. "Generalizing Multi-Agent Path Finding for Heterogeneous Agents." en. *Proceedings of the International Symposium on Combinatorial Search*, 11, 1, 101–105. Number: 1. doi:10.1609/socs.v11i1.18540.

P. Bachor, R.-D. Bergdoll, and B. Nebel. June 2023. "The Multi-Agent Transportation Problem." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 10, (June 2023), 11525–11532. doi:10.1609/aaai.v37i10.26362.

Y. Bai, S. Kotpalliwar, C. Kanellakis, and G. Nikolakopoulos. Feb. 2025. "Multi-agent Path Planning Based on Conflict-Based Search (CBS) Variations for Heterogeneous Robots." en. *Journal of Intelligent & Robotic Systems*, 111, 1, (Feb. 2025), 26. doi:10.1007/s10846-025-02229-0.

M. Barer, G. Sharon, R. Stern, and A. Felner. July 2014. "Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem." en. In: *Seventh Annual Symposium on Combinatorial Search*. (July 2014). Retrieved Dec. 12, 2022 from https://www.aaai.org/ocs/index.php/SOCS/SOCS14/paper/view/8911.

R. Barták, M. Ivanová, and J. Švancara. Apr. 2021a. "Colored Multi-Agent Path Finding: Solving Approaches." en. *The International FLAIRS Conference Proceedings*, 34, (Apr. 2021). doi:10.32473/flairs.v34i1.128535.

R. Barták, M. Ivanová, and J. Švancara. July 2021b. "From Classical to Colored Multi-Agent Path Finding." en. *Proceedings of the International Symposium on Combinatorial Search*, 12, 1, (July 2021), 150–152. Number: 1. doi:10.1609/socs.v12i1.18566.

R. Barták and J. Mestek. May 2021. "OzoMorph: Demonstrating Colored Multi-Agent Path Finding on Real Robots." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 18, (May 2021), 15991–15993. Number: 18. doi:10.1609/aaai.v35i18.17990.

G. Belov, W. Du, M. Garcia De La Banda, D. Harabor, S. Koenig, and X. Wei. Sept. 2021. "From Multi-Agent Pathfinding to 3D Pipe Routing." en. *Proceedings of the International Symposium on Combinatorial Search*, 11, 1, (Sept. 2021), 11–19. doi:10.1609/socs.v11i1.18530.

J. van den Berg and M. Overmars. Aug. 2005. "Prioritized motion planning for multiple robots." In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. (Aug. 2005), 430–435. doi:10.1109/IROS.2005.1545306.

A. Berndt, N. van Duijkeren, L. Palmieri, A. Kleiner, and T. Keviczky. 2024. "Receding Horizon Re-Ordering of Multi-Agent Execution Schedules." *IEEE Transactions on Robotics*, 40, 1356–1372. Conference Name: IEEE Transactions on Robotics. doi:10.1109/TRO.2023.3344051.

A. Berndt, N. Van Duijkeren, L. Palmieri, and T. Keviczky. Oct. 2020. *A Feedback Scheme to Reorder a Multi-Agent Execution Schedule by Persistently Optimizing a Switchable Action Dependency Graph*. en. arXiv:2010.05254 [cs]. (Oct. 2020). Retrieved Apr. 4, 2023 from http://arxiv.org/abs/2010.05254.

L. Bonalumi, B. Flammini, D. Azzalini, and F. Amigoni. Apr. 2025. "Multi-Agent Pickup and Delivery with external agents." *Robotics and Autonomous Systems*, (Apr. 2025), 105000. doi:10.1016/j.robot.2025.105000.

K. Brown, O. Peltzer, M. A. Sehr, M. Schwager, and M. J. Kochenderfer. May 2020. "Optimal Sequential Task Assignment and Path Finding for Multi-Agent Robotic Assembly Planning." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. (May 2020), 441–447. doi:10.1109/ICRA40945.2020.9197527.

H.-D. Bui, E. Plaku, and G. J. Stein. Feb. 2025. *Multi-Agent Path Finding under Limited Communication Range Constraint via Dynamic Leading*. arXiv:2501.02770 [cs]. (Feb. 2025). doi:10.48550/arXiv.2501.02770.

S.-H. Chan, Z. Chen, T. Guo, H. Zhang, Y. Zhang, D. Harabor, S. Koenig, C. Wu, and J. Yu. June 2024. "The League of Robot Runners Competition: Goals, Designs, and Implementation." en. In: (June 2024). Retrieved Feb. 13, 2025 from https://openreview.net/forum?id=mPmCnEHTvJ.

R. Chandra, R. Maligi, A. Anantula, and J. Biswas. June 2023. "SocialMapf: Optimal and Efficient Multi-Agent Path Finding With Strategic Agents for Social Navigation." *IEEE Robotics and Automation Letters*, 8, 6, (June 2023), 3214–3221. Conference Name: IEEE Robotics and Automation Letters. doi:10.1109/LRA.2023.3265169.

Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey. July 2021. "Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery." en. *IEEE Robotics and Automation Letters*, 6, 3, (July 2021), 5816–5823. doi:10.1109/LRA.2021.3074883.

Z. Chen, D. Harabor, J. Li, and P. J. Stuckey. June 2024. "Traffic Flow Optimisation for Lifelong Multi-Agent Path Finding (Extended Abstract)." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 265–266. doi:10.1609/socs.v17i1.31573.

Z. Chen, D. D. Harabor, J. Li, and P. J. Stuckey. May 2021. "Symmetry Breaking for k-Robust Multi-Agent Path Finding." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 14, (May 2021), 12267–12274. Number: 14. doi:10.1609/aaai.v35i14.17456.

S. Choudhury, K. Solovey, M. J. Kochenderfer, and M. Pavone. Jan. 2021. *Efficient Large-Scale Multi-Drone Delivery Using Transit Networks*. en. arXiv:1909.11840 [cs]. (Jan. 2021). Retrieved Jan. 4, 2023 from http://arxiv.org/abs/1909.11840.

M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. Apr. 2021. "PRIMAL2: Pathfinding via Reinforcement and Imitation Multi-Agent Learning – Lifelong." en. *IEEE Robotics and Automation Letters*, 6, 2, (Apr. 2021), 2666–2673. arXiv:2010.08184 [cs]. doi:10.1109/LRA.2021.3062803.

S. Dergachev and K. Yakovlev. Aug. 2024. *Decentralized Unlabeled Multi-agent Pathfinding Via Target And Priority Swapping (With Supplementary)*. arXiv:2408.14948 [cs]. (Aug. 2024). doi:10.48550/arXiv.2408.14948.

K. Dresner and P. Stone. Mar. 2008. "A Multiagent Approach to Autonomous Intersection Management." en. *Journal of Artificial Intelligence Research*, 31, (Mar. 2008), 591–656. doi:10.1613/jair.2502.

E. Ewing, J. Ren, D. Kansara, V. Sathiyanarayanan, and N. Ayanian. May 2022. "Betweenness Centrality in Multi-Agent Path Finding." In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (AAMAS '22). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2022), 400–408. ISBN: 978-1-4503-9213-6. Retrieved Jan. 3, 2023 from.

A. Felner and R. Stern. Mar. 2026. "Multi-Agent Path Finding with Unassigned Agents (MAPFUA)." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40, 46, (Mar. 2026), 39691–39698. doi:10.1609/aaai.v40i46.41322.

A. Felner, R. Stern, S. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek. Sept. 2021. "Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges." en. *Proceedings of the International Symposium on Combinatorial Search*, 8, 1, (Sept. 2021), 29–37. Number: 1. doi:10.1609/socs.v8i1.18423.

Y. Feng, A. Paul, Z. Chen, and J. Li. May 2024. "A Real-Time Rescheduling Algorithm for Multi-robot Plan Execution." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34, (May 2024), 201–209. doi:10.1609/icaps.v34i1.31477.

G. Fine, D. Atzmon, and N. Agmon. 2023. "Anonymous Multi-Agent Path Finding with Individual Deadlines." In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* (AAMAS '23). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 869–877. ISBN: 978-1-4503-9432-1. Retrieved Dec. 30, 2024 from.

F. Fioravantes, D. Knop, J. M. Křišťan, N. Melissinos, and M. Opler. Mar. 2026. "Exact Algorithms for Multiagent Path Finding with Communication Constraints on Tree-Like Structures." en. *Journal of Artificial Intelligence Research*, 85, (Mar. 2026). doi:10.1613/jair.1.19325.

J. Gao, Y. Li, X. Li, K. Yan, K. Lin, and X. Wu. Jan. 2024. "A review of graph-based multi-agent pathfinding solvers: From classical to beyond classical." *Knowledge-Based Systems*, 283, (Jan. 2024), 111121. doi:10.1016/j.knosys.2023.111121.

J. Gao, Y. Li, Y. Mu, Q. Liu, H. Chen, and Y. Lou. 2025. "CBTMP: Optimizing Multi-Agent Path Finding in Heterogeneous Cooperative Environments." *IEEE Robotics and Automation Letters*, 1–8. doi:10.1109/LRA.2025.3557672.

T. Geft. July 2023. "Fine-Grained Complexity Analysis of Multi-Agent Path Finding on 2D Grids." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 20–28. Number: 1. doi:10.1609/socs.v16i1.27279.

B. P. Gerkey and M. J. Matarić. Sept. 2004. "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems." EN. *The International Journal of Robotics Research*, 23, 9, (Sept. 2004), 939–954. Publisher: SAGE Publications Ltd STM. doi:10.1177/0278364904045564.

N. Greshler, O. Gordon, O. Salzman, and N. Shimkin. Nov. 2021. "Cooperative Multi-Agent Path Finding: Beyond Path Planning and Collision Avoidance." In: *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. (Nov. 2021), 20–28. doi:10.1109/MRS50823.2021.9620590.

C. Henkel, J. Abbenseth, and M. Toussaint. Nov. 2019. "An Optimal Algorithm to Solve the Combined Task Allocation and Path Finding Problem." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. (Nov. 2019), 4140–4146. doi:10.1109/IROS40897.2019.8968096.

J. Herynek and S. Edelkamp. Dec. 2024. *Heuristic Planner for Communication-Constrained Multi-Agent Multi-Goal Path Planning*. arXiv:2412.13719 [cs]. (Dec. 2024). doi:10.48550/arXiv.2412.13719.

F. Ho, R. Geraldes, A. Gonçalves, B. Rigault, B. Sportich, D. Kubo, M. Cavazza, and H. Prendinger. Feb. 2022. "Decentralized Multi-Agent Path Finding for UAV Traffic Management." *IEEE Transactions on Intelligent Transportation Systems*, 23, 2, (Feb. 2022), 997–1008. Conference Name: IEEE Transactions on Intelligent Transportation Systems. doi:10.1109/TITS.2020.3019397.

F. Ho, A. Goncalves, A. Salta, M. Cavazza, R. Geraldes, and H. Prendinger. May 2019. *Multi-agent path finding for UAV traffic management: Robotics track*. en. Conference Proceedings. ISBN: 9781450363099 ISSN: 2523-5699 Num Pages: 2457 Pages: 131-139 Publisher: Association for Computing Machinery (ACM). (May 2019). Retrieved Dec. 18, 2024 from https://dl.acm.org/doi/10.5555/3306127.3331684.

W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. July 2018. "Conflict-Based Search with Optimal Task Assignment." In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (July 2018), 757–765. Retrieved Jan. 18, 2023 from.

W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. Apr. 2019. "Persistent and Robust Execution of MAPF Schedules in Warehouses." en. *IEEE Robotics and Automation Letters*, 4, 2, (Apr. 2019), 1125–1131. doi:10.1109/LRA.2019.2894217.

W. Hönig, T. K. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig. Mar. 2016. "Multi-Agent Path Finding with Kinematic Constraints." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26, (Mar. 2016), 477–485. doi:10.1609/icaps.v26i1.13796.

W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian. Aug. 2018. "Trajectory Planning for Quadrotor Swarms." *IEEE Transactions on Robotics*, 34, 4, (Aug. 2018), 856–869. Conference Name: IEEE Transactions on Robotics. doi:10.1109/TRO.2018.2853613.

W. Hönig, T. Satish Kumar, H. Ma, S. Koenig, and N. Ayanian. Oct. 2016. "Formation change for robot groups in occluded environments." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. (Oct. 2016), 4836–4842. doi:10.1109/IROS.2016.7759710.

M. Husár, J. Švancara, P. Obermeier, R. Barták, and T. Schaub. May 2022. "Reduction-based Solving of Multi-agent Pathfinding on Large Maps Using Graph Pruning." In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (AAMAS '22). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2022), 624–632. ISBN: 978-1-4503-9213-6. Retrieved Jan. 16, 2023 from.

M. Ivanová and P. Surynek. 2013. "Adversarial cooperative path-finding: a first view." In: *Proceedings of the 17th AAAI Conference on Late-Breaking Developments in the Field of Artificial Intelligence* (AAAIWS'13-17). AAAI Press, 53–55. Retrieved Dec. 30, 2024 from.

M. Ivanová and P. Surynek. Nov. 2014. "Adversarial Cooperative Path-Finding: Complexity and Algorithms." In: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. ISSN: 2375-0197. (Nov. 2014), 75–82. doi:10.1109/ICTAI.2014.22.

M. Ivanová and P. Surynek. Nov. 2021. "Adversarial Multi-Agent Path Finding is Intractable." en. In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, Washington, DC, USA, (Nov. 2021), 481–486. ISBN: 978-1-6654-0898-1. doi:10.1109/ICTAI52525.2021.00078.

M. Ivanová, P. Surynek, and K. Hirayama. Jan. 2018. "Area Protection in Adversarial Path-finding Scenarios with Multiple Mobile Agents on Graphs - A Theoretical and Experimental Study of Strategies for Defense Coordination." en. In: *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018)*. Vol. 2. SCITEPRESS, (Jan. 2018), 184–191. ISBN: 978-989-758-275-2. doi:10.5220/0006583601840191.

K. Janovská and P. Surynek. 2021. "Hierarchical Control of Swarms during Evacuation." en. In: *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. SCITEPRESS - Science and Technology Publications, Online Streaming, — Select a Country —, 61–73. ISBN: 978-989-758-533-3. doi:10.5220/0010678200003064.

K. Janovská and P. Surynek. Sept. 2024. *Multi-agent Path Finding in Continuous Environment*. arXiv:2409.10680 [cs]. (Sept. 2024). doi:10.48550/arXiv.2409.10680.

H. Jiang, M. Lin, and J. Li. Apr. 2025. "Speedup Techniques for Switchable Temporal Plan Graph Optimization." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39, 22, (Apr. 2025), 23212–23221. Number: 22. doi:10.1609/aaai.v39i22.34487.

H. Jiang, Y. Wang, R. Veerapaneni, T. Duhan, G. Sartoretti, and J. Li. Oct. 2024. *Deploying Ten Thousand Robots: Scalable Imitation Learning for Lifelong Multi-Agent Path Finding*. arXiv:2410.21415 [cs]. (Oct. 2024). doi:10.48550/arXiv.2410.21415.

H. Jiang, Y. Zhang, R. Veerapaneni, and J. Li. June 2024. "Scaling Lifelong Multi-Agent Path Finding to More Realistic Settings: Research Challenges and Opportunities." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 234–242. doi:10.1609/socs.v17i1.31565.

Y. Jo and H. I. Son. May 2024. "Field Evaluation of a Prioritized Path-Planning Algorithm for Heterogeneous Agricultural Tasks of Multi-UGVs." In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. (May 2024), 11891–11897. doi:10.1109/ICRA57147.2024.10610857.

O. Kaduri, E. Boyarski, and R. Stern. June 2020. "Algorithm Selection for Optimal Multi-Agent Pathfinding." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30, (June 2020), 161–165. doi:10.1609/icaps.v30i1.6657.

K. Kasaura, R. Yonetani, and M. Nishimura. June 2023. "Periodic Multi-Agent Path Planning." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 5, (June 2023), 6183–6191. Number: 5. doi:10.1609/aaai.v37i5.25762.

J. Kottinger, T. Geft, S. Almagor, O. Salzman, and M. Lahijanian. June 2024. "Introducing Delays in Multi Agent Path Finding." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 37–45. doi:10.1609/socs.v17i1.31540.

N. M. Kou, C. Peng, H. Ma, T. K. S. Kumar, and S. Koenig. Apr. 2020. "Idle Time Optimization for Target Assignment and Path Finding in Sortation Centers." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 06, (Apr. 2020), 9925–9932. Number: 06. doi:10.1609/aaai.v34i06.6547.

E. Lam, P. J. Stuckey, and D. Harabor. Dec. 2024. "Optimal Multi-Agent Pickup and Delivery Using Branch-and-Cut-and-Price Algorithms." *Transportation Science*, (Dec. 2024). Publisher: INFORMS. doi:10.1287/trsc.2023.0268.

V. Lehoux-Lebacque, T. Silander, C. Loiodice, S. Lee, A. Wang, and S. Michel. 2024. "Multi-Agent Path Finding with Real Robot Dynamics and Interdependent Tasks for Automated Warehouses." en. In: *ECAI 2024*. IOS Press, 4393–4401. doi:10.3233/FAIA241017.

B. Li and H. Ma. June 2023. "Double-Deck Multi-Agent Pickup and Delivery: Multi-Robot Rearrangement in Large-Scale Warehouses." *IEEE Robotics and Automation Letters*, 8, 6, (June 2023), 3701–3708. Conference Name: IEEE Robotics and Automation Letters. doi:10.1109/LRA.2023.3272272.

J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig. June 2022. "MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36, 9, (June 2022), 10256–10265. Number: 9. doi:10.1609/aaai.v36i9.21266.

J. Li, Z. Chen, Y. Zheng, S.-H. Chan, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig. May 2021. "Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31, (May 2021), 477–485. doi:10.1609/icaps.v31i1.15994.

J. Li, D. Harabor, P. J. Stuckey, H. Ma, G. Gange, and S. Koenig. Dec. 2021. "Pairwise symmetry reasoning for multi-agent path finding search." en. *Artificial Intelligence*, 301, (Dec. 2021), 103574. doi:10.1016/j.artint.2021.103574.

J. Li, K. Sun, H. Ma, A. Felner, T. K. S. Kumar, and S. Koenig. 2020. "Moving Agents in Formation in Congested Environments." In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '20). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 726–734. ISBN: 978-1-4503-7518-4. Retrieved Nov. 17, 2025 from.

J. Li, P. Surynek, A. Felner, H. Ma, T. K. S. Kumar, and S. Koenig. July 2019. "Multi-Agent Path Finding for Large Agents." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 01, (July 2019), 7627–7634. Number: 01. doi:10.1609/aaai.v33i01.33017627.

J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig. May 2021. "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 13, (May 2021), 11272–11281. Number: 13. doi:10.1609/aaai.v35i13.17344.

J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto. Dec. 2024. *Multi-Agent Path Finding in Continuous Spaces with Projected Diffusion Models*. en. arXiv:2412.17993 [cs]. (Dec. 2024). doi:10.48550/arXiv.2412.17993.

M. Liu, H. Ma, J. Li, and S. Koenig. 2019. "Task and Path Planning for Multi-Agent Pickup and Delivery." In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '19). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1152–1160. ISBN: 978-1-4503-6309-9. Retrieved Feb. 9, 2025 from.

G. Lodigiani, N. Basilico, and F. Amigoni. Sept. 2023. "Robust Multi-Agent Pickup and Delivery with Delays." In: *2023 European Conference on Mobile Robots (ECMR)*. ISSN: 2767-8733. (Sept. 2023), 1–8. doi:10.1109/ECMR59166.2023.10256381.

H. Ma. Sept. 2022. "Graph-Based Multi-Robot Path Finding and Planning." en. *Current Robotics Reports*, 3, 3, (Sept. 2022), 77–84. doi:10.1007/s43154-022-00083-8.

H. Ma and S. Koenig. Oct. 2017. "AI buzzwords explained: multi-agent path finding (MAPF)." en. *AI Matters*, 3, 3, (Oct. 2017), 15–19. doi:10.1145/3137574.3137579.

H. Ma and S. Koenig. May 2016. "Optimal Target Assignment and Path Finding for Teams of Agents." In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (AAMAS '16). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2016), 1144–1152. ISBN: 978-1-4503-4239-1. Retrieved Jan. 16, 2023 from.

H. Ma, S. Koenig, et al.. Feb. 2017. *Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios*. arXiv:1702.05515 [cs]. (Feb. 2017). doi:10.48550/arXiv.1702.05515.

H. Ma, T. K. S. Kumar, and S. Koenig. Feb. 2017. "Multi-Agent Path Finding with Delay Probabilities." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31, 1, (Feb. 2017). Number: 1. doi:10.1609/aaai.v31i1.11035.

H. Ma, J. Li, T. S. Kumar, and S. Koenig. May 2017. "Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks." In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '17). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2017), 837–845. Retrieved Jan. 17, 2023 from.

H. Ma, C. Tovey, G. Sharon, T. K. Kumar, and S. Koenig. Mar. 2016. "Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30, 1, (Mar. 2016). Number: 1. doi:10.1609/aaai.v30i1.10409.

H. Ma, G. Wagner, A. Felner, J. Li, T. K. S. Kumar, and S. Koenig. 2018. "Multi-agent path finding with deadlines." In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (IJCAI'18). AAAI Press, Stockholm, Sweden, 417–423. ISBN: 978-0-9992411-2-7. Retrieved Dec. 30, 2024 from.

H. Ma, J. Yang, L. Cohen, T. K. S. Kumar, and S. Koenig. Sept. 2017. "Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments." en. In: *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*. (Sept. 2017). Retrieved Dec. 12, 2022 from https://www.aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15845.

N. Madar, K. Solovey, and O. Salzman. July 2022. "Leveraging Experience in Lifelong Multi-Agent Pathfinding." en. *Proceedings of the International Symposium on Combinatorial Search*, 15, 1, (July 2022), 118–126. Number: 1. doi:10.1609/socs.v15i1.21759.

T. Matsui. Feb. 2025. "Investigation of MAPF Problem Considering Fairness and Worst Case." en. In: vol. 2. SCITEPRESS, (Feb. 2025), 224–234. ISBN: 978-989-758-737-5. doi:10.5220/0013389600003890.

J. Morag, N. Gabay, D. Koyfman, and R. Stern. July 2025. "Should Multi-Agent Path Finding Algorithms Coordinate Target Arrival Times?" en. *Proceedings of the International Symposium on Combinatorial Search*, 18, (July 2025), 231–235. doi:10.1609/socs.v18i1.35998.

J. Morag, R. Stern, and A. Felner. July 2023. "Adapting to Planning Failures in Lifelong Multi-Agent Path Finding." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 47–55. Number: 1. doi:10.1609/socs.v16i1.27282.

R. Morris, C. S. Pasareanu, K. Luckow, W. Malik, H. Ma, T. K. S. Kumar, and S. Koenig. Mar. 2016. "Planning, Scheduling and Monitoring for Airport Surface Operations." en. In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*. (Mar. 2016). Retrieved Jan. 14, 2023 from https://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12611.

G. Mouratidis, B. Nebel, and S. Koenig. June 2024. "Fools Rush in Where Angels Fear to Tread in Multi-Goal CBS." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 243–251. doi:10.1609/socs.v17i1.31566.

A. Natan, R. Stern, and M. Kalech. 2023. "Blame Attribution for Multi-Agent Path Finding Execution Failures." en. In: *ECAI 2023*. IOS Press, 1763–1770. doi:10.3233/FAIA230462.

M. Nekvinda and R. Barták. Nov. 2021. "Contingent Planning for Robust Multi-Agent Path Finding." In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. ISSN: 2375-0197. (Nov. 2021), 487–492. doi:10.1109/ICTAI52525.2021.00079.

V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, and W. Yeoh. Aug. 2017. "Generalized target assignment and path finding using answer set programming." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (IJCAI'17). AAAI Press, Melbourne, Australia, (Aug. 2017), 1216–1223. ISBN: 978-0-9992411-0-3. Retrieved Jan. 14, 2023 from.

A. Okoso, K. Otaki, S. Koide, and T. Nishi. Oct. 2022. "High Density Automated Valet Parking via Multi-agent Path Finding." In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. (Oct. 2022), 2146–2153. doi:10.1109/ITSC55140.2022.9922035.

A. Okoso, K. Otaki, and T. Nishi. Oct. 2019. "Multi-Agent Path Finding with Priority for Cooperative Automated Valet Parking." In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. (Oct. 2019), 2135–2140. doi:10.1109/ITSC.2019.8917112.

K. Okumura. May 2024. "Engineering LaCAM*: Towards Real-time, Large-scale, and Near-optimal Multi-agent Pathfinding." In: *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems* (AAMAS '24). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2024), 1501–1509. ISBN: 979-8-4007-0486-4. Retrieved Feb. 28, 2025 from.

K. Okumura. June 2023. "LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 10, (June 2023), 11655–11662. doi:10.1609/aaai.v37i10.26377.

K. Okumura, F. Bonnet, Y. Tamura, and X. Défago. Aug. 2023. "Offline Time-Independent Multiagent Path Planning." *IEEE Transactions on Robotics*, 39, 4, (Aug. 2023), 2720–2737. Conference Name: IEEE Transactions on Robotics. doi:10.1109/TRO.2023.3258690.

K. Okumura and X. Défago. Aug. 2023. "Solving simultaneous target assignment and path planning efficiently with time-independent execution." *Artificial Intelligence*, 321, (Aug. 2023), 103946. doi:10.1016/j.artint.2023.103946.

K. Okumura, M. Machida, X. Défago, and Y. Tamura. Sept. 2022. "Priority inheritance with backtracking for iterative multi-agent path finding." *Artificial Intelligence*, 310, (Sept. 2022), 103752. doi:10.1016/j.artint.2022.103752.

K. Okumura, Y. Tamura, and X. Défago. May 2021. "Time-Independent Planning for Multiple Moving Agents." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 13, (May 2021), 11299–11307. Number: 13. doi:10.1609/aaai.v35i13.17347.

K. Okumura and S. Tixeuil. June 2023. "Fault-Tolerant Offline Multi-Agent Path Planning." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 10, (June 2023), 11647–11654. Number: 10. doi:10.1609/aaai.v37i10.26376.

O. Peltzer, K. Brown, M. Schwager, M. J. Kochenderfer, and M. Sehr. Aug. 2021. *STT-CBS: A Conflict-Based Search Algorithm for Multi-Agent Path Finding with Stochastic Travel Times*. arXiv:2004.08025 [cs]. (Aug. 2021). doi:10.48550/arXiv.2004.08025.

X. Peng, O. Simonin, and C. Solnon. Oct. 2023. "Non-Crossing Anonymous MAPF for Tethered Robots." en. *Journal of Artificial Intelligence Research*, 78, (Oct. 2023), 357–384. doi:10.1613/jair.1.14351.

A. A. H. Qizilbash, C. Henkel, and S. Mostaghim. June 2020. "Ant Colony Optimization based Multi-Robot Planner for Combined Task Allocation and Path Finding." In: *2020 17th International Conference on Ubiquitous Robots (UR)*. ISSN: 2325-033X. (June 2020), 487–493. doi:10.1109/UR49135.2020.9144944.

J. Ren, V. Sathiyanarayanan, E. Ewing, B. Senbaslar, and N. Ayanian. May 2021. "MAPFAST: A Deep Algorithm Selector for Multi Agent Path Finding using Shortest Path Embeddings." In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '21). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2021), 1055–1063. ISBN: 978-1-4503-8307-3. Retrieved Jan. 18, 2023 from.

Z. Ren, Y. Cai, and H. Wang. Oct. 2024. "Multi-Agent Teamwise Cooperative Path Finding and Traffic Intersection Coordination." In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. (Oct. 2024), 7990–7995. doi:10.1109/IROS58592.2024.10802803.

Z. Ren, J. Li, H. Zhang, S. Koenig, S. Rathinam, and H. Choset. July 2023. "Binary Branching Multi-Objective Conflict-Based Search for Multi-Agent Path Finding." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33, (July 2023), 361–369. doi:10.1609/icaps.v33i1.27214.

Z. Ren, A. Nandy, S. Rathinam, and H. Choset. Sept. 2024. "DMS*: Towards Minimizing Makespan for Multi-Agent Combinatorial Path Finding." *IEEE Robotics and Automation Letters*, 9, 9, (Sept. 2024), 7987–7994. doi:10.1109/LRA.2024.3436333.

Z. Ren, S. Rathinam, and H. Choset. 2024. "A Bounded Sub-Optimal Approach for Multi-Agent Combinatorial Path Finding." *IEEE Transactions on Automation Science and Engineering*, 1–16. Conference Name: IEEE Transactions on Automation Science and Engineering. doi:10.1109/TASE.2024.3466183.

Z. Ren, S. Rathinam, and H. Choset. Apr. 2023a. "A Conflict-Based Search Framework for Multiobjective Multiagent Path Finding." *IEEE Transactions on Automation Science and Engineering*, 20, 2, (Apr. 2023), 1262–1274. doi:10.1109/TASE.2022.3183183.

Z. Ren, S. Rathinam, and H. Choset. June 2022. "Conflict-Based Steiner Search for Multi-Agent Combinatorial Path Finding." en. In: *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, (June 2022). ISBN: 978-0-9923747-8-5. doi:10.15607/RSS.2022.XVIII.058.

Z. Ren, S. Rathinam, and H. Choset. May 2021a. "MS: A New Exact Algorithm for Multi-agent Simultaneous Multi-goal Sequencing and Path Finding." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. (May 2021), 11560–11565. doi:10.1109/ICRA48506.2021.9561779.

Z. Ren, S. Rathinam, and H. Choset. May 2021b. "Multi-objective Conflict-based Search for Multi-agent Path Finding." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. (May 2021), 8786–8791. doi:10.1109/ICRA48506.2021.9560985.

Z. Ren, S. Rathinam, and H. Choset. July 2023b. "Search Algorithms for Multi-Agent Teamwise Cooperative Path Finding [Extended Abstract]." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 181–182. doi:10.1609/socs.v16i1.27304.

Z. Ren, S. Rathinam, and H. Choset. Oct. 2021c. "Subdimensional Expansion for Multi-Objective Multi-Agent Path Finding." *IEEE Robotics and Automation Letters*, 6, 4, (Oct. 2021), 7153–7160. Conference Name: IEEE Robotics and Automation Letters. doi:10.1109/LRA.2021.3096744.

T. Shahar, S. Shekhar, D. Atzmon, A. Saffidine, B. Juba, and R. Stern. Mar. 2021. "Safe Multi-Agent Pathfinding with Time Uncertainty." en. *Journal of Artificial Intelligence Research*, 70, (Mar. 2021), 923–954. doi:10.1613/jair.1.12397.

Y. Shaoul, R. Veerapaneni, M. Likhachev, and J. Li. June 2024. "Unconstraining Multi-Robot Manipulation: Enabling Arbitrary Constraints in ECBS with Bounded Sub-Optimality." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 109–117. doi:10.1609/socs.v17i1.31548.

G. Sharon, R. Stern, A. Felner, and N. Sturtevant. July 2012. "Conflict-based search for optimal multi-agent path finding." In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (AAAI'12). AAAI Press, Toronto, Ontario, Canada, (July 2012), 563–569. Retrieved Dec. 26, 2022 from.

Y. Sherma, E. Weiss, and O. Salzman. July 2025. "From Agent Centric to Obstacle Centric Planning: A Makespan-Optimal Algorithm for the Multi-Agent Warehouse Rearrangement Problem." en. *Proceedings of the International Symposium on Combinatorial Search*, 18, (July 2025), 136–144. doi:10.1609/socs.v18i1.35985.

B. Shofer, G. Shani, and R. Stern. July 2023. "Multi Agent Path Finding under Obstacle Uncertainty." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33, (July 2023), 402–410. doi:10.1609/icaps.v33i1.27219.

D. Sigurdson, V. Bulitko, S. Koenig, C. Hernandez, and W. Yeoh. June 2019. *Automatic Algorithm Selection In Multi-agent Pathfinding*. arXiv:1906.03992 [cs]. (June 2019). doi:10.48550/arXiv.1906.03992.

D. Sigurdson, V. Bulitko, W. Yeoh, C. Hernández, and S. Koenig. Aug. 2018. "Multi-Agent Pathfinding with Real-Time Heuristic Search." In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. ISSN: 2325-4289. (Aug. 2018), 1–8. doi:10.1109/CIG.2018.8490436.

A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov. Aug. 2023. "Learn to Follow: Lifelong Multi-agent Pathfinding with Decentralized Replanning." en. In: (Aug. 2023). Retrieved Jan. 11, 2025 from https://openreview.net/forum?id=vMV4tsA639.

T. Standley. July 2010. "Finding Optimal Solutions to Cooperative Pathfinding Problems." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24, 1, (July 2010), 173–178. Number: 1. doi:10.1609/aaai.v24i1.7564.

R. Stern. 2019. "Multi-Agent Path Finding – An Overview." en. In: *Artificial Intelligence*. Vol. 11866. Ed. by G. S. Osipov, A. I. Panov, and K. S. Yakovlev. Series Title: Lecture Notes in Computer Science. Springer International Publishing, Cham, 96–115. ISBN: 978-3-030-33273-0 978-3-030-33274-7. doi:10.1007/978-3-030-33274-7_6.

R. Stern et al.. July 2019. "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks." en. In: *Twelfth Annual Symposium on Combinatorial Search*. (July 2019). Retrieved Jan. 17, 2023 from https://aaai.org/ocs/index.php/SOCS/SOCS19/paper/view/18341.

K. J. Strawn, T. Phan, E. Wang, N. Ayanian, S. Koenig, and L. Lindemann. July 2025. *Multi-Agent Path Finding Among Dynamic Uncontrollable Agents with Statistical Safety Guarantees*. en. arXiv:2507.22282 [cs]. (July 2025). doi:10.48550/arXiv.2507.22282.

Y. Su, R. Veerapaneni, and J. Li. Mar. 2024. "Bidirectional Temporal Plan Graph: Enabling Switchable Passing Orders for More Efficient Multi-Agent Path Finding Plan Execution." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38, 16, (Mar. 2024), 17559–17566. Number: 16. doi:10.1609/aaai.v38i16.29706.

P. Surynek. July 2010. "An Optimization Variant of Multi-Robot Path Planning Is Intractable." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24, 1, (July 2010), 1261–1263. Number: 1. doi:10.1609/aaai.v24i1.7767.

P. Surynek. 2021a. "Continuous Multi-agent Path Finding via Satisfiability Modulo Theories (SMT)." en. In: *Agents and Artificial Intelligence*. Vol. 12613. Ed. by A. P. Rocha, L. Steels, and J. van den Herik. Series Title: Lecture Notes in Computer Science. Springer International Publishing, Cham, 399–420. ISBN: 978-3-030-71157-3 978-3-030-71158-0. doi:10.1007/978-3-030-71158-0_19.

P. Surynek. May 2021b. "Multi-Goal Multi-Agent Path Finding via Decoupled and Integrated Goal Vertex Ordering." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 14, (May 2021), 12409–12417. Number: 14. doi:10.1609/aaai.v35i14.17472.

P. Surynek. July 2022. "Problem Compilation for Multi-Agent Path Finding: a Survey." en. In: *Electronic proceedings of IJCAI 2022*. Vol. 6. ISSN: 1045-0823. (July 2022), 5615–5622. doi:10.24963/ijcai.2022/783.

P. Surynek. 2019. "Unifying Search-Based and Compilation-Based Approaches to Multi-Agent Path Finding through Satisfiability Modulo Theories." en. *Proceedings of the International Symposium on Combinatorial Search*, 10, 1, 202–203. Number: 1. doi:10.1609/socs.v10i1.18491.

P. Surynek, A. Felner, R. Stern, and E. Boyarski. Aug. 2016. "Efficient SAT approach to multi-agent path finding under the sum of costs objective." In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence* (ECAI'16). IOS Press, NLD, (Aug. 2016), 810–818. ISBN: 978-1-61499-671-2. doi:10.3233/978-1-61499-672-9-810.

P. Surynek, A. Felner, R. Stern, and E. Boyarski. 2018. "Sub-Optimal SAT-Based Approach to Multi-Agent Path-Finding Problem." en. *Proceedings of the International Symposium on Combinatorial Search*, 9, 1, 99–105. Number: 1. doi:10.1609/socs.v9i1.18456.

P. Surynek, Y. Zheng, E. Kline, S. Koenig, and T. K. Satish Kumar. Oct. 2024. "Virtual Network Embedding as Boolean Satisfiability." In: *2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI)*. ISSN: 2375-0197. (Oct. 2024), 387–395. doi:10.1109/ICTAI62512.2024.00063.

J. Švancara, D. Atzmon, K. Strauch, R. Kaminski, and T. Schaub. 2024. "Which Objective Function is Solved Faster in Multi-Agent Pathfinding? It Depends." en. In: *Proceedings of the 16th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, Rome, Italy, 23–33. ISBN: 978-989-758-680-4. doi:10.5220/0012249400003636.

J. Švancara and R. Barták. 2019. "Combining Strengths of Optimal Multi-Agent Path Finding Algorithms." en. In: *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, Prague, Czech Republic, 226–231. ISBN: 978-989-758-350-6. doi:10.5220/0007470002260231.

J. Švancara and R. Barták. Jan. 2023. "Tackling Train Routing via Multi-agent Pathfinding and Constraint-based Scheduling." en. In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, (Jan. 2023), 306–313. ISBN: 978-989-758-547-0. doi:10.5220/0010869700003116.

J. Švancara, E. Tignon, R. Barták, T. Schaub, P. Wanko, and R. Kaminski. July 2023. "Multi-Agent Pathfinding with Predefined Paths: To Wait, or Not to Wait, That Is the Question [Extended Abstract]." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 185–186. Number: 1. doi:10.1609/socs.v16i1.27306.

J. Švancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták. July 2019. "Online Multi-Agent Pathfinding." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 01, (July 2019), 7732–7739. Number: 01. doi:10.1609/aaai.v33i01.33017732.

J. Švancara, D. Zahrádka, M. Subramanian, R. Barták, and M. Kulich. 2025. "Generating Safe Policies for Multi-Agent Path Finding with Temporal Uncertainty." en. In: *Proceedings of the 17th International Conference on Agents and Artificial Intelligence*. Vol. 3. SCITEPRESS, 1238–1244. ISBN: ISBN 978-989-758-737-5.

W. J. Tan, X. Tang, and W. Cai. May 2024. "Robust Multi-Agent Pathfinding with Continuous Time." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34, (May 2024), 570–578. doi:10.1609/icaps.v34i1.31519.

Y. Tang, S. Koenig, and J. Li. June 2024. "ITA-ECBS: A Bounded-Suboptimal Algorithm for Combined Target-Assignment and Path-Finding Problem." en. *Proceedings of the International Symposium on Combinatorial Search*, 17, (June 2024), 134–142. doi:10.1609/socs.v17i1.31551.

Y. Tang, Z. Ren, J. Li, and K. Sycara. Dec. 2023. "Solving Multi-Agent Target Assignment and Path Finding with a Single Constraint Tree." In: *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. (Dec. 2023), 8–14. doi:10.1109/MRS60187.2023.10416794.

Y. Tang, X. Xiong, J. Xi, J. Li, E. Bıyık, and S. Koenig. July 2025. "RAILGUN: A Unified Convolutional Policy for Multi-Agent Path Finding Across Different Environments and Tasks (Extended Abstract)." en. *Proceedings of the International Symposium on Combinatorial Search*, 18, (July 2025), 273–274. doi:10.1609/socs.v18i1.36015.

Y. Tang, Z. Yu, Y. Zheng, T. K. S. Kumar, J. Li, and S. Koenig. Jan. 2025. *Enhancing Lifelong Multi-Agent Path Finding with Cache Mechanism.* arXiv:2501.02803 [cs]. (Jan. 2025). doi:10.48550/arXiv.2501.02803.

S. Thomas, D. Deodhare, and M. N. Murty. Nov. 2015. "Extended Conflict-Based Search for the Convoy Movement Problem." *IEEE Intelligent Systems*, 30, 6, (Nov. 2015), 60–70. Conference Name: IEEE Intelligent Systems. doi:10.1109/MIS.2015.96.

D. Vainshtein, Y. Sherma, K. Solovey, and O. Salzman. July 2023. "Terraforming – Environment Manipulation during Disruptions for Multi-Agent Pickup and Delivery." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 92–100. Number: 1. doi:10.1609/socs.v16i1.27287.

T. T. Walker, N. R. Sturtevant, and A. Felner. July 2018. "Extended Increasing Cost Tree Search for Non-Unit Cost Domains." en. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, (July 2018), 534–540. ISBN: 978-0-9992411-2-7. doi:10.24963/ijcai.2018/74.

Q. Wan, C. Gu, S. Sun, M. Chen, H. Huang, and X. Jia. Nov. 2018. "Lifelong Multi-Agent Path Finding in A Dynamic Environment." In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. (Nov. 2018), 875–882. doi:10.1109/ICARCV.2018.8581181.

H. Wang and W. Chen. Apr. 2022. "Multi-Robot Path Planning With Due Times." *IEEE Robotics and Automation Letters*, 7, 2, (Apr. 2022), 4829–4836. doi:10.1109/LRA.2022.3152701.

J. Wang, J. Li, H. Ma, S. Koenig, and S. Kumar. Jan. 2019. "A New Constraint Satisfaction Perspective on Multi-Agent Path Finding: Preliminary Results." en. *18th International Conference on Autonomous Agents and Multi-Agent Systems*, (Jan. 2019). Retrieved Dec. 21, 2022 from https://par.nsf.gov/biblio/10118766-new-constraint-satisfaction-perspective-multi-agent-path-finding-preliminary-results.

Q. Wang, R. Veerapaneni, Y. Wu, J. Li, and M. Likhachev. May 2024. "MAPF in 3D Warehouses: Dataset and Analysis." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34, (May 2024), 623–632. doi:10.1609/icaps.v34i1.31525.

L. Wen, Y. Liu, and H. Li. Apr. 2022. "CL-MAPF: Multi-Agent Path Finding for Car-Like robots with kinematic and spatiotemporal constraints." en. *Robotics and Autonomous Systems*, 150, (Apr. 2022), 103997. doi:10.1016/j.robot.2021.103997.

X. Wu, Y. Liu, X. Tang, W. Cai, F. Bai, G. Khonstantine, and G. Zhao. 2022. "Multi-Agent Pickup and Delivery with Task Deadlines." In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (WI-IAT '21). Association for Computing Machinery, New York, NY, USA, 360–367. ISBN: 978-1-4503-9115-3. doi:10.1145/3486622.3493915.

P. R. Wurman, R. D'Andrea, and M. Mountz. Mar. 2008. "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses." en. *AI Magazine*, 29, 1, (Mar. 2008), 9–9. Number: 1. doi:10.1609/aimag.v29i1.2082.

H. Xiong, S. Shi, D. Ren, and J. Hu. June 2022. "A survey of job shop scheduling problem: The types and models." *Computers & Operations Research*, 142, (June 2022), 105731. doi:10.1016/j.cor.2022.105731.

Q. Xu, J. Li, S. Koenig, and H. Ma. Oct. 2022. "Multi-Goal Multi-Agent Pickup and Delivery." In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. (Oct. 2022), 9964–9971. doi:10.1109/IROS47612.2022.9981785.

K. Yakovlev and A. Andreychuk. June 2017. "Any-Angle Pathfinding for Multiple Agents Based on SIPP Algorithm." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 27, (June 2017), 586–594. doi:10.1609/icaps.v27i1.13856.

K. Yakovlev, A. Andreychuk, T. Rybecký, and M. Kulich. July 2020. "On the Application of Safe-Interval Path Planning to a Variant of the Pickup and Delivery Problem." en. In: vol. 2. SCITEPRESS, (July 2020), 521–528. ISBN: 978-989-758-442-8. doi:10.5220/0009888905210528.

K. Yakovlev, A. Andreychuk, and V. Vorobyev. Sept. 2019. "Prioritized Multi-agent Path Finding for Differential Drive Robots." In: *2019 European Conference on Mobile Robots (ECMR)*. (Sept. 2019), 1–6. doi:10.1109/ECMR.2019.8870957.

T. Yamauchi, Y. Miyashita, and T. Sugawara. Dec. 2022. "Efficient Path and Action Planning Method for Multi-Agent Pickup and Delivery Tasks under Environmental Constraints." en. *SN Computer Science*, 4, 1, (Dec. 2022), 83. doi:10.1007/s42979-022-01475-5.

J. Yan and J. Li. Apr. 2025. "Multi-Agent Motion Planning for Differential Drive Robots Through Stationary State Search." en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39, 22, (Apr. 2025), 23360–23368. Number: 22. doi:10.1609/aaai.v39i22.34503.

R. Yoeli, R. Stern, and D. Atzmon. Sept. 2021. "Measuring the Vulnerability of a Multi-Agent Pathfinding Solution." en. *Proceedings of the International Symposium on Combinatorial Search*, 10, 1, (Sept. 2021), 208–209. doi:10.1609/socs.v10i1.18494.

J. Yu and S. M. LaValle. 2013. "Multi-agent Path Planning and Network Flow." en. In: *Algorithmic Foundations of Robotics X*. Vol. 86. Ed. by E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus. Series Title: Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, Berlin, Heidelberg, 157–173. ISBN: 978-3-642-36278-1 978-3-642-36279-8. doi:10.1007/978-3-642-36279-8_10.

D. Zahrádka, A. Andreychuk, M. Kulich, and K. Yakovlev. Jan. 2022. "Quality Analysis of Multi-Agent Multi-Item Pickup and Delivery Solutions Using a Decoupled Approach." en. In: *IFAC-PapersOnLine*. Vol. 55. (Jan. 2022), 61–66. doi:10.1016/j.ifacol.2023.01.134.

D. Zahrádka, D. Kubišta, and M. Kulich. 2023. "Solving Robust Execution of Multi-Agent Pathfinding Plans as a Scheduling Problem." *Planning and Robotics, ICAPS'23 Workshop*.

D. Zahrádka, D. Mužíková, M. Kulich, J. Švancara, and R. Barták. 2025. "Towards Holistic Approach to Robust Execution of MAPF Plans." en. In: *Proceedings of the 17th International Conference on Agents and Artificial Intelligence*. Vol. 1. SCITEPRESS - Science and Technology Publications, Porto, Portugal, 624–631. ISBN: 978-989-758-737-5. doi:10.5220/0013319700003890.

H. Zhang, J. Chen, J. Li, B. C. Williams, and S. Koenig. May 2022. "Multi-Agent Path Finding for Precedence-Constrained Goal Sequences." In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (AAMAS '22). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (May 2022), 1464–1472. ISBN: 978-1-4503-9213-6. Retrieved Jan. 14, 2023 from.

Y. Zhang, X. Wu, H. Wang, and Z. Ren. Sept. 2024. *Multi-Agent Combinatorial Path Finding with Heterogeneous Task Duration.* arXiv:2311.15330 [cs]. (Sept. 2024). doi:10.48550/arXiv.2311.15330.

Y. Zhang, Z. Chen, D. Harabor, P. L. Bodic, and P. J. Stuckey. May 2024. "Planning and Execution in Multi-Agent Path Finding: Models and Algorithms." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34, (May 2024), 707–715. doi:10.1609/icaps.v34i1.31534.

Y. Zhang, D. Harabor, P. L. Bodic, and P. J. Stuckey. July 2023. "Efficient Multi Agent Path Finding with Turn Actions." en. *Proceedings of the International Symposium on Combinatorial Search*, 16, 1, (July 2023), 119–127. Number: 1. doi:10.1609/socs.v16i1.27290.

Y. Zheng, S. Ravi, E. Kline, S. Koenig, and T. K. S. Kumar. June 2022. "Conflict-Based Search for the Virtual Network Embedding Problem." en. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32, (June 2022), 423–433. doi:10.1609/icaps.v32i1.19828.

Y. Zheng, S. Ravi, E. Kline, L. Thurlow, S. Koenig, and T. K. S. Kumar. July 2023. "Improved Conflict-Based Search for the Virtual Network Embedding Problem." In: *2023 32nd International Conference on Computer Communications and Networks (ICCCN).* ISSN: 2637-9430. (July 2023), 1–10. doi:10.1109/ICCCN58024.2023.10230188.

X. Zhong, J. Li, S. Koenig, and H. Ma. May 2022. "Optimal and Bounded-Suboptimal Multi-Goal Task Assignment and Path Finding." en. In: *2022 International Conference on Robotics and Automation (ICRA).* IEEE, Philadelphia, PA, USA, (May 2022), 10731–10737. ISBN: 978-1-7281-9681-7. doi:10.1109/ICRA46639.2022.9812020.

Received 28 November 2025; accepted 04 May 2026