Blue-Sky: Multi-Agent Path Finding with Unassigned Agents (MAPFUA)

Ariel Felner and Roni Stern

Faculty of Computer and Information Science
Institute of the Foundations of Artificial Intelligence
Ben-Gurion University of the Negev, Israel
felner@bgu.ac.il, roni.stern@gmail.com

Abstract

In the Multi-Agent Path Finding (MAPF) problem, the aim is to find collision free paths for multiple agents. MAPF has many practical applications and has spawned massive research interest in the past two decades. Most MAPF research assumed that every agent is assigned a target it must reach. This assumption often does not hold in several key applications such as automated warehouses and parking lots, where some agents are assigned targets to reach, and others, denoted as unassigned agents, can either stay idle or move to clear the way for the assigned agents. In this paper we introduce this important problem, explain its uniqueness and encourage the entire community to work on it.

1 Introduction and overview

In the Multi-Agent Path Finding problem (MAPF) (Stern et al. 2019), the aim is to find collision-free paths for multiple mobile agents (robots, vehicles, people etc.). MAPF has applications in traffic control, aviation, robotic warehouses etc. Consequently, MAPF spawned massive research recently with thousands of published papers, including many papers by the authors. Numerous variants of MAPF were presented over the years. Perhaps the most prominent variant (received a large attention starting around 2010) is the offline (one shot) version where the input receives start and target locations for the agents and the output is a set of paths, one for each agent from its start to its target. A more realistic variant of MAPF that captured researchers a few years later is that of Lifelong MAPF where agents are repeatedly assigned targets after they reaches their current targets.

Nevertheless, despite the wide research on MAPF, a common assumption shaped this research problem to be more scientific but less applicative. The assumption is that every agent is assigned a target and it must follow a path from its start to its target. This assumption often does not hold in several key applications such as automated warehouses and parking lots, where some agents are assigned targets to reach, and others, denoted as *unassigned agents*, have no targets to go to and can either stay idle or move to clear the way for the assigned agents.

In this paper we introduce the problem of *Multi-agent Path Finding with Unassigned Agents* (MAPFUA), where *assigned agents* have targets to go to while *unassigned agents* do not have targets but are allowed to move to clear the way for the assigned agents. We define a number of variants and associated costs functions for MAPFUA including: **(1)** fastest arrival of the assigned agents to their targets, **(2)** minimizing Fuel consumption and **(3)** minimizing the number of unassigned agents that move. We then explain the applicability of MAPFUA and provide a number of research directions to pursue. Finally we provide a number of representative results that demonstrate the vast richness of this problem. Thus, researchers are encouraged to work on this practical and fascinating problem which generalizes MAPF.

2 Existing research on MAPF

The input to MAPF is a graph G; a set of k agents $a_1, \ldots, a_k$; and a start and target location for each agent, denoted $s_1, \ldots, s_k$ and $t_1, \ldots, t_k$, respectively. The task is to find a path for each agent from its start to its target such that no two agents conflict, i.e., they never occupy the same location at the same time step. In some settings, conflicts also include agents crossing the same edge in opposite directions at the same time. A primary application of MAPF is in automated warehouses (Li et al. 2021b), where the products are placed on shelves and robots are repeatedly assigned tasks to pick items from the shelves and bring them to some delivery station for shipping out to the clients (see `www.youtube.com/watch?v=qshXW4zEUPc`). MAPF has many other practical applications in factories, digital entertainment, traffic control, aviation, and more. Therefore, in the last two decades MAPF attracted significant interest from scientists and practitioners in diverse areas such as Artificial Intelligence, Robotics, Industrial Engineering etc. A variety of settings and cost objectives (i.e., metrics to minimize) have been proposed, along with numerous algorithms designed to solve these problems (see our surveys in (Felner et al. 2017; Stern et al. 2019)). MAPF is very challenging (proved NP-hard (Yu and LaValle 2013b; Nebel 2020; Surynek 2010)) due to the exponential number of applicable actions per state (=mapping of all agents to locations), as well as the exponential number of states.

2.1 Classical MAPF

The classical variant of MAPF described above is called the Offline (one shot) MAPF variant as the output is a plan

that includes a non-conflicting path for each of the agents. One approach to address it is to apply combinatorial search algorithms such as A* enhanced by MAPF-specific techniques (Standley 2010a; Goldenberg et al. 2014), or compile it to different NP-Hard problems such as SAT and ASP and then use dedicated solvers (Surynek et al. 2016; Erdem et al. 2013). A different approach is to plan for each agent separately and invoke conflict-resolution methods to coordinate these plans. Primary examples of algorithms that follow this approach are Prioritized Planning (PP) (Silver 2005), Large Neighborhood Search for MAPF (MAPF-LNS) (Li et al. 2022, 2021a), the Increasing Cost Tree Search (ICTS) (Sharon et al. 2013), and the Conflict-based Search algorithm (CBS) (Sharon et al. 2015). Yet another approach for solving MAPF problems is to plan for all agents together but limit the planning horizon. Prominent examples of this approach include PIBT (Okumura et al. 2022), LaCAM (Okumura 2023), and PIE (Zhang et al. 2024). Our lab in BGU (managed jointly by the authors) developed key algorithms from each of these approaches (Sharon et al. 2013, 2015; Felner et al. 2012; Surynek et al. 2016), including CBS, which is considered state-of-the-art for optimally solving MAPF, received several awards (Sharon et al. 2015, 2012), and has spawned many variants and enhancements (Barer et al. 2014; Zhang et al. 2022; Felner et al. 2018; Boyarski et al. 2021) inter alia.

Contemporary research on MAPF focuses on the following directions: (i) Developing new algorithms for classical MAPF sometimes for various objectives to minimize such as Sum-of-Costs (Sharon et al. 2015), Makespan (Maliah, Atzmon, and Felner 2025) and Fuel (Koyfman et al. 2025). (ii) Developing suboptimal and incomplete algorithms for MAPF with the aim to find solutions faster and for larger problems (with many more agents) but sacrificing solution optimality and sometimes even the completeness of the solving algorithm. (iii) Exploring different settings of MAPF such as continuous environments (Andreychuk et al. 2022), agents with different shapes (Li et al. 2019), Uncertainty (Wagner and Choset 2017; Atzmon et al. 2020; Shofer, Shani, and Stern 2023) and Privacy (Keskin et al. 2024).

2.2 Lifelong MAPF

Lifelong MAPF (LMAPF) is another variant of MAPF that was spawned in the past 15 years (albeit later than classical MAPF). LMAPF better suits the online nature of many of the practical MAPF applications such as warehouses and parking lots. In LMAPF when an agent reaches its target it receives a new target to go to. The objective of a LMAPF algorithm is to repeatedly find conflict-free paths for the agents aiming to maximize the system throughput, which is the number of times the agents reached their targets. In warehouses, system throughput corresponds to the number of packages delivered over time. A basic algorithm for LMAPF (which has dozens of variants and improvements) is based on repeatedly solving the MAPF problem based on the current set of locations of the agents and the current set of targets. To speed up the search and consider the uncertainty over future targets, it is common to apply the Rolling Horizon Collision Resolution (RHCR) mechanism

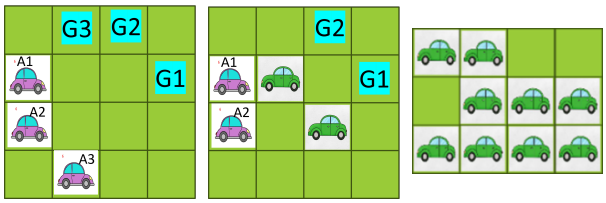

Figure 1: MAPF (left). MAPFUA (mid). alr (Right)

(Li et al. 2021b). In RHCR, we plan for all agents while ignoring conflicts that are expected to occur after a predefined number of time steps referred to as the planning horizon. Then, we execute the prefix of this plan and repeat this planning procedure as needed. Recent work proposed interleaving planning and execution, planning for the next sequence of actions while performing them (Zhang et al. 2024).

2.3 Goal Allocation in MAPF

In classical MAPF every agent is assigned a unique target. Extensions of MAPF also exist. In Anonymous MAPF (AMAPF) agents are interchangeable and can be assigned to any target. Finding makespan optimal solutions for AMAPF can be done in polynomial time (Yu and LaValle 2013a) while optimizing the sum-of-costs is much harder (Honig et al. 2018). In the Multi-Agent Combinatorial Path Finding (MCPF) problem, each agent is assigned a unique target but the returned paths must also visit additional targets on their way to their assigned target (Ren, Rathinam, and Choset 2021, 2023).

3 MAPF with Unassigned Agents

A common prominent assumption that exists in almost all of the work on MAPF is that every agent is assigned a target (possibly more targets in MCPF). While this setting is appealing and spawned massive research as surveyed above, we believe that it is relatively far from the reality of many key applications that were mentioned by many authors for justifications of the applicability of their work.

We next consider a novel generalization of MAPF which we believe is much more suitable to many real-world applications. In the new problem some agents remain *unassigned*, i.e., they do not have a target they must reach, but may still be able to move to make room for the other agents. We call this problem **MAPF with Unassigned Agents (MAPFUA)**. The agents in MAPFUA are divided into two classes: **(1) Assigned agents**, agents that are assigned targets to go to; and **(2) Unassigned agents**, which do not have a target, yet still occupy a location and can optionally move if this is helpful. A solution to MAPFUA is a path for each of the assigned agents from its start location to its target location, as well as paths for the unassigned agents. The paths for the unassigned agents may be trivial – stay in place, but may involve moving to clear the path for the assigned agents if this is helpful for improving the solution (e.g., throughput of the system or other metrics, see below). Figure 1(left) shows a classic MAPF problem instance while figure 1(mid) shows a MAPFUA problem instance where blue agents are assigned agents while green agents are unassigned agents.

3.1 Applicability of MAPFUA

MAPFUA is directly suitable for warehouses where some of the robots do not have tasks (e.g., when there are currently more robots than tasks). Similarly, in LMAPF, when an agent reaches its target, its task is completed. If a new task is not yet assigned to it, then it should be treated as an unassigned agent and may move away from its (already reached) target if this is helpful for other agents (we recently highlighted this (Pertzovsky et al. 2025)). Traditional automated warehouses were not densely populated with delivery robots and packages. Thus, most of the time all robots were assigned tasks to perform and existing MAPF algorithms sufficed. Modern automated warehouses, however, are huge and include thousands of robots in them. Such environments often include agents with no specific task assigned to it.

Additionally, MAPFUA is most suitable in warehouses where there are no carrying robots but all packages have moving abilities. For example, consider a 3D grid (or any other graph) where some cells occupy packages with merchandise. The task is to navigate some of these packages along the grid to specific target locations (e.g., docking stations). Mechanically, packages can move if we install some kind of robotic wheels for them, or, alternatively, packages are placed on a structure of moving tracks and can be moved anywhere via these tracks. In such warehouses, each package is now a mobile entity, logically a (cheap) robot. Naturally, packages without targets can be treated as unassigned agents and be optionally moved from their locations if this is helpful for other agents.

Such warehouses already exist in modern storage automation systems including: Autonomously Moving Packages, Robotic Parking Lots and Automated Greenhouses for plants that move on demand. See for example:
(1) `versatileautomation.com/why-versatile`,
(2) `silmanautomatedparkingsystems.com`,
(3) `www.roboticparking.com`,
(4) `parkplusinc.com`,
(5) `www.wps.eu/nl/tuinbouw/buffersystemen`,
(6) `www.greenhousemag.com/article/gm0112-movable-tray-systems`.

All these examples exactly exhibit MAPFUA: some packages, cars or plants must be moved to users while others must be moved away to make way.

Despite all these real world applications, MAPFUA received minimal attention from the research community. We thus believe that after so many years that the research community spent on the classical version of MAPF it should move and focus on this new variant. The contributions of relevant algorithms and research on such problems are sorely needed in the relevant industry just mentioned.

3.2 Formal Definition of MAPFUA

Formally, MAPFUA is defined by a graph $G = (V, E)$, a set of *assigned agents* $A = \{a_1, \ldots, a_n\}$, and a set of *unassigned agents* $U = \{u_1, \ldots, u_m\}$. Each assigned agent $a_i \in A$ is associated with a start vertex $s_i \in V$ and a target vertex $t_i \in V$, while each unassigned agent $u_j \in U$ is associated with a start vertex $s_j \in V$ but no target. A solution π to this problem is a mapping of each agent (assigned or unassigned) to a path in G, where for each assigned agent $a_i \in A$, the path π_i starts at s_i and ends at t_i, and for each unassigned agent $u_j \in U$, the path π_j starts at s_j and may end at any vertex in V. The paths of all agents, whether assigned or unassigned, must be conflict-free.

We next define practical types of MAPFUA and characterize them according to different solution criteria and cost functions. We demonstrate these types of MAPFUA on a parking-lot use case where car owners come to pick up their cars (these cars are the assigned agents). Human employees must drive cars to their owners but are also allowed to move the other cars (unassigned agents) to clear the way in case they are blocking the assigned agents. Of course, other practical examples are evident too.

3.3 Solution criteria.

A natural solution criterion for MAPFUA (taken from MAPF) is that all assigned agents end up at their targets without conflicts along the way. Other solution criteria are also be practical in MAPFUA. For example, in the parking lot use case it is sufficient for each car to reach its owner, but not necessarily at the same time. This is known as the *reachability criterion* (Okumura et al. 2022; Morag et al. 2025). Another solution criterion is where each agent has a set of goals that it can choose from. In the parking-lot, this corresponds to cases where the cars are moved from the owner to some position in the parking lot. It does not matter which position in the lot is chosen as long as it is vacant. This is somewhat related to Anonymized MAPF (Yu and LaValle 2013a; Honig et al. 2018) and to MCPF (Ren, Rathinam, and Choset 2021, 2023), but here, there might be more targets than (assigned) agents and not all targets must be visited.

4 Cost Functions for MAPFUA

We next define serval possible cost functions for MAPFUA and again demonstrate them on the parking-lot use case.

(1) Sum of Service Time (SST). This cost is the sum of time steps required to move each assigned agent to its target. SST corresponds maximizing the system throughput.

(2) Fuel. This cost is the total number of moves made by either assigned or unassigned agents (and waiting does not cost). The Fuel cost corresponds to minimizing the energy costs required to move the agents.

(3) Number of Unassigned Agents that Move (NUA). Here we do not care about the lengths of the paths but aim to minimize the number of unassigned agents that move. In the parking lot example, entering a car and starting its engine is costly and should be minimized for two reasons. First, owners of the unassigned cars are not happy when their cars are moved (fuel is lost and an accident may occur). Second, the human employees work harder when entering a new car (you have to find the keys, remember the codes etc.)

Figure 2 illustrates the differences between these solution costs. In this example (top left, a), the car at the top left corner (assigned agent, blue) needs to be moved to the owner at the top right. Figure 2b shows the optimal solution for

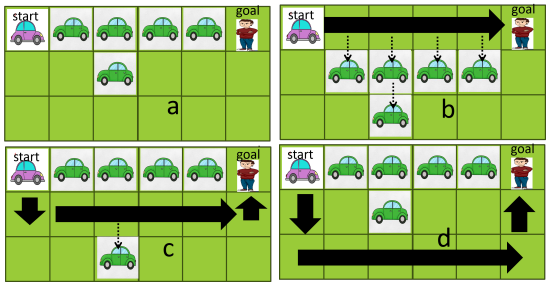

Figure 2: MAPFUA instance (a) and its cost functions (b–d)

minimizing SST: all unassigned cars (green) move one step down at time step t_1 (dashed arrows) and then the assigned car moves all the way to the right and reaches the target at time step t_6. Figure 2c shows the solution for minimizing Fuel: only one unassigned car moves down and then the assigned car goes down one row, then right all the way, and finally up to the target (total of 8 moves). Finally, Figure 2d shows the solution for minimizing NUA: the assigned car bypasses all other car, and no other car is moved. Recently, we worked on a variant of Fuel where there is a single assigned agent and only its moves are counted but moves of unassigned agents are not counted (Pertzovsky, Stern, and Zivan 2024a; Pertzovsky et al. 2025).

5 Challenges and Research Directions

The introduction of MAPFUA spawns many challenges and calls for new research directions as we next list (this is of course not an exhaustive list). We encourage the entire MAPF community to take part in these directions.

5.1 Characterization of MAPFUA

Since MAFPUA has rarely been studied, the first task would be to theoretically characterize it, find its key variants and evaluate the computational complexity of optimizing the different cost function. The influence of the number of unassigned agents relative to the number of assigned agents and relative to the size of the environment should be studied.

5.2 Optimal Algorithms for MAPFUA

The next task would be to develop (optimal) algorithms for the various MAPFUA variants and cost functions, based on existing single- and multi-agent path finding algorithms. For example, adapting A* to MAPFUA is relatively simple by using the common *joint state-space* for MAPF where states include all the permutations of agents into locations. At each time step, every agent can either stay idle (wait) or move to an adjacent location (move). Naturally, agents are allowed to move in parallel. The joint state-space can be optimized further for MAPFUA, as any permutation of the m unassigned agents into the same set of m locations is equivalent. Naturally, the costs of the different actions depend on the different cost functions mentioned above. Next, for an A*-based system, admissible heuristics should be developed for the various cost functions. Finally, many enhancements and improvement that were suggested for classical MAPF (such as independence detection and operator decomposition (Standley 2010b), EPEA* (Goldenberg et al. 2014)) etc.) should be adapted to A*-based MAPFUA solvers.

Another research direction can migrate CBS to MAPFUA. A regular CBS tree is developed. A goal node is where all assigned agents have paths to their targets while unassigned agents have any possible path. Conflicts between assigned agents are treated in the classic way. Conflicts that include unassigned agents are challenging because an unassigned agent does not have a target. Therefore, in its low level, an unassigned agent needs to find a path (that goes somewhere) that satisfies the required constraint, and heuristics should be developed to guide the corresponding search.

5.3 Suboptimal algorithms for MAPFUA

Practical applications often prefer to trade-off solution cost for runtime, in order to ensure timely response time to end users. Such algorithms should be developed for MAPFUA. A general approach for this is to use Bounded Suboptimal Search (BSS) algorithms such as *Weighted A** (Pohl 1973) and Explicit Estimation Search (EES) (Thayer and Ruml 2011). BSS algorithms accept a suboptimality bound B and return a solution with cost $\leq B \times OPT$, where OPT is the optimal solution cost. This approach should be followed also for MAPFUA, building on the optimal algorithms suggested above. Our prior work on BSS for MAPF yielded the ECBS algorithm (Barer et al. 2014), which combines CBS and Focal search. Adapting ECBS and its more sophisticated extensions like EECBS (Li, Ruml, and Koenig 2021) to MAPFUA is another possible direction. Of course, the many other algorithms (besides the few examples just given) developed for MAPF should also be modified for MAPFUA. Many non-trivial challenges will arise here and we again encourage the community to take part in this endeavor.

5.4 MAPFUA in Dense Environments

Many of the MAPFUA use cases (e.g., parking lots and warehouses) are in environments that are incentivized to utilized space. Consequently, such environments often include regions that are densely populated with robots/agents. Dense environments are becoming more common as the use of mobile agents increases, in particular in modern warehouses.

In dense regions, moving unassigned agents to clear the way for the assigned agents is sometimes extremely useful and even crucial. Let alr be the ratio between the number of agents and the total number of locations that agents may occupy. In Figure 1(right), $alr = 9/12$. Intuitively, a MAPF environment is dense if alr is close to 1 (very few empty cells). The option of moving unassigned agents in sparse environments (with small alr) is not that significant. In such environments treating unassigned agents as static obstacle (and not moving them at all) is a reasonable policy because assigned agents can simply bypass the unassigned agents.[1]

[1]Of course, doing this may lose solution optimality. Moreover, if an unassigned agent is blocking an entrance to a region that contains the target then completeness can also be forfeited. Indeed, it will be easy to draw synthetic examples where optimality and even completeness are forfeited. However, such examples are very rare

By contrast, dense environments (with large alr) are particularly important for MAPFUA because bypasses of assigned agents are harder to perform or even impossible. The ability to move unassigned agents can make a big difference on the runtime of the algorithm, on the quality of the solution that is returned and on whether or not the algorithm is complete.

Thus, developing algorithms for MAPFUA with particular interest in dense environments is essential. Again, this should be done while taking all the cost functions defined above into consideration. Additionally, both optimal- and suboptimal algorithms should be developed specifically this case. Furthermore, the definition based on $alr \approx 1$ is rather simple as other aspects of the environment such as its topology may also be crucial for identifying algorithm failure. Thus, a deeper research on what makes the environment dense is needed. Additionally, environments are sometimes mixed with both sparse and dense regions. For example, large warehouses have densely populated areas, e.g., near the picking stations, while other areas are very sparse (denoted by sparse-dense environments). Thus, treatment of such environments also requires deep studying.

6 Combining MAPFUA in Other Settings

Another important challenge is to add the notion of unassigned agents to other non-classical settings of MAPF. We list a few of them here with specific focus on recently introduced MAPF settings. We encourage researchers to work on these setting as well, ideally in combination with the MAPFUA idea of having unassigned agents.

6.1 MAPFUA for Lifelong Setting

A viable research direction is to incorporate the above techniques of MAPFUA into lifelong setting, where new tasks are repeatedly allocated to agents over time. Such setting raise algorithmic challenges such as how to interleave planning and execution, how to optimize the system behavior over time, and how to dynamically detect and treat dense and sparse regions to ensure high throughput.

6.2 MAPF with the Reachability Criterion

The classical solution criterion of MAPF is that given a start and a target configuration of the agents the task is to move the agents from the start to the target configuration. That is, we need a specific time step where all agents are in their goal configuration (at the same time). The vast majority of the research on MAPF used this criterion. Recently, Morag et al. (2025) importantly argued that in many realistic settings there is only a need for an agent to *arrive* at its target (as soon as possible). But an agent is then free to leave its target and there is no need to see all agents located in their targets at the same time. Such variant is called MAPF with the reachability criterion (MAPFRO). A very important example that was given by Morag et al. (2025) is of Lifelong MAPF (LMAPF) where an agent receives a new target as soon as it arrives at its current target. This is certainly the case in modern warehouses where after delivering a package there is no need for the agent to stay at its target and it immediately plans its path to its new target. But, in practice, many LMAPF algorithms continue to use the classical criterion when solving LMAPF.

We would like to point out that the introduction of MAPFUA and of unassigned agents is directly relevant for MAPFRO. MAPFRO can be reduced to MAPFUA as follows. After reaching its target (until it receives a new target) an agent becomes an unassigned agent. Thus, the algorithm can choose to either move such an agent or to leave it idle in its current position. Alternatively, MAPFUA can be reduced to MAPFRO by setting the start location of an unassigned agent as its target. Thus, at time t_0 the agent has arrived at its target and is free to move away.

6.3 The Multi-Agent Warehouse Rearrangement

A new variant of MAPF called the Multi-Agent Warehouse Rearrangement problem (MAWR) was introduced very recently (Sherma, Weiss, and Salzman 2025). Again, the input is a start and a target configuration of agents. The agents, referred to as *moving obstacles*, must all travel from their initial configuration to their goal configuration. However, the agents cannot move on their own and a carrying robot must carry them along their intended path. This problem is of course very realistic and is evident in many modern warehouses such as Amazon etc. (See video above). They formalized MAWR as a general search problem in which the output is a set of paths for the carrying robots that transfer all the agents from their initial configuration to their final configuration. They also introduced a CBS-based algorithm that solves this problem optimally for the makespan cost function. Their algorithm has two phases: the first phase plans paths for the agents. The second phase assigns the carrying robots to move these agents along their paths.

An important and interesting direction is to expand the fascinating MAWR problem by adding the notion of unassigned agents. Indeed, in realistic warehouses not all items need be moved. So, now the task is to assign the robots to move the assigned agents to their targets. These robots can also be instructed to move unassigned agents if moving them will improve the the cost of the overall solution. Here too, variants of all our cost functions described above are valid and could be very relevant to real world applications.

7 Preliminary Experimental Results

Our group has started to work on some of the directions listed above. Here we present a taste of these preliminary directions and some representative experimental results for the three cost functions described above.

7.1 STT of CBS with Unassigned Agents

We evaluated three policies for handling unassigned agents within CBS (Sharon et al. 2015), which is a prominent optimal algorithm. **(1)** CBS_baseline treats unassigned agents as static obstacles that can not be moved. **(2)** CBS solves a classic MAPF problem, where the unassigned agents are assigned their start locations as their targets. That is, they are

in practice and for many practical use cases which are sparse with agents, treating them as static obstacles will suffice.

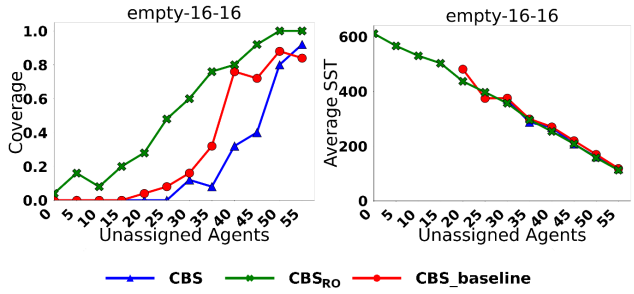

Figure 3: Coverage. Varying the # of unassigned agents.

allowed to move away but must return to their start states. **(3)** CBS_RO assumes that the unassigned agents do not have targets and can thus move away as needed.

Figure 3 show experimental results where a total 60 agents were present, and the number of unassigned agents was varied. The x-axis shows the number of unassigned agents (out of the 60 agents), and the y-axis shows coverage when given 1 minute per instance (left) or average SST (right). As can be seen, the coverage of all CBS variants increase when more agents become unassigned. This is reasonable because *all* the three policies for unassigned agents have less planning to do than for regular assigned agents. In CBS_RO (green) the unassigned agents have the larger degree of freedom, and it has the best coverage among all variants. Classic CBS allows the unassigned agents to move but forces them to later return to their start states, while CBS_baseline treats them as obstacles. Thus, it seems that Classic CBS (Blue) gives more freedom to the unassigned agents than CBS_baseline (red). But, surprisingly, CBS_baseline has a better coverage, probably because CBS must create a non-conflicting path for each of these agents and this demands time.

The average SST per agent is given in Figure 3(right). All variants have similar SST and this is attributed to the fact that the environment was relatively sparse ($alr = 60/256 = 0.23$). Thus, all assigned agents found relative similar cost paths, despite the fact that some of them had more time consuming computation. We conjecture that for denser environments the STT will be vary among the different polices.

Further exploring all these issues is an example of the wide research field that is ahead of us.

7.2 MAPFUA with a Single Assigned Agent

Recently, we considered a MAPFUA case with a single assigned agent. This is motivated by emergency vehicles, e.g., an ambulance, that need to move as smooth as possible to the hospital while all other vehicles move to clear the way. For this case we developed the Corridor Generating Algorithm (CGA) (Pertzovsky, Stern, and Zivan 2024b), a polynomial-time rule-based algorithm guaranteed to find a solution if one exists under general assumptions. CGA is designed to minimize *fuel* of the assigned agent. It is an open question how to optimize time for the assigned agent, or how to minimize the movements of all other agents. We later generzlied CGA to the case a multiple assigned agents and intoduced the Multi-agent CGA algorithm (Pertzovsky et al. 2025).

Agents	Max	IH	MH
2/14	13,539 (4.9s)	12,186 (**5.7s**)	**12,119** (6.7s)
2/16	162,477 (89s)	59,313 (**26s**)	**56,883** (32s)
2/18	136,565 (73s)	51,428 (22s)	**30,124** (**18s**)
2/20	125,886 (55s)	47,062 (19s)	**17,501** (**14s**)

Table 1: Nodes expanded and runtime (in secs) on a 5×5 grid with varying numbers of unassigned agents.

7.3 Optimally solving MAPFUA for NUA

We implemented a variant of A* for the NUA cost (minimize the number of unassigned agents that move) using the *joint agent state-space*. For heuristics we have done the following. For each assigned agent a_i we performed a secondary search where a_i may step into the location of an unassigned agent a_j and in this case a_j is deleted from the environment (similar to the *capture* action in chess). The aim of this secondary search is to find a path for a_i to its target while minimizing the number of unassigned agents that are captured (denoted $cap(a_i)$). $cap(a_i)$ is an admissible heuristic for agent a_i. We then implemented three ways to aggregate $cap(a_i)$ among all assigned agents. **(1) Max**. Here we took the maximum of $cap(a_i)$ over all assigned agents. **(2) Increasing Heuristics (IH)**. Here, we iterate over all individual unassigned agents and check whether their individual removal will allow all assigned agents to get the their targets. If so, IH is set to 1. Otherwise we do the same for all pairs, then triples etc. **(3) Merge Heuristic (MH)**. Here, we *merge* together all assigned agents into a joint-agent (meta-agent) and search for this joint-agent for the minimal number of captures. Intuitively, the latter heuristics incur larger overhead per node but are more informed.

Table 1 presents the number of nodes expanded (and times in seconds) on a 5×5 grid with $x = 2$ assigned agents and y unassigned agents, denoted in the table by x/y. Indeed the stronger heuristics caused a significant reduction in the number of nodes expanded. Timing results showed similar trends but naturally the stronger heuristics incurs more overhead per node. For timing, IH is most suitable for cases with small number of unassigned agent while MH is most suitable for cases with a small number of assigned agents. We aim to further investigate this phenomenon.

8 Summary and Conclusions

We have introduced the MAPFUA problem (including a number of possible cost functions) and explained its importance and its applicability. We presented several direction for future research on MAPFUA. Finally, we included a number of preliminary results that demonstrate the richness of MAPFUA. We strongly encourage the large MAPF research community to take part in researching this fascinating, more-general and more-applicative problem.

9 Acknowledgments

We sincerely thank Jonathan Morag, Noy Gabai, Bar Dolev and Arseniy Pertzovsky, all students on our lab, who performed the experiments reported in this paper.

References

Andreychuk, A.; Yakovlev, K. S.; Surynek, P.; Atzmon, D.; and Stern, R. 2022. Multi-agent pathfinding with continuous time. *Artif. Intell.*, 305: 103662.

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N.-F. 2020. Robust multi-agent path finding and executing. *JAIR*, 67: 549–579.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. In *SoCS*, 19–27.

Boyarski, E.; Felner, A.; Bodic, P. L.; Harabor, D. D.; Stuckey, P. J.; and Koenig, S. 2021. f-Aware Conflict Prioritization & Improved Heuristics For Conflict-Based Search. In *AAAI*, 12241–12248.

Erdem, E.; Kisa, D.; Oztok, U.; and Schüller, P. 2013. A general formal framework for pathfinding problems with multiple agents. In *AAAI Conference on Artificial Intelligence*, 290–296.

Felner, A.; Goldenberg, M.; Sharon, G.; Stern, R.; Beja, T.; Sturtevant, N. R.; Schaeffer, J.; and Holte, R. 2012. Partial-Expansion A* with Selective Node Generation. In *AAAI*, 471–477.

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2018. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In *ICAPS*, 83–87.

Felner, A.; Stern, R.; Shimony, S. E.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N. R.; Wagner, G.; and Surynek, P. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *SoCS*, 29–33.

Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N. R.; Holte, R. C.; and Schaeffer, J. 2014. Enhanced Partial Expansion A. *J. Artif. Intell. Res.*, 50: 141–187.

Honig, W.; Kiesel, S.; Tinka, A.; Durham, J.; and Ayanian, N. 2018. Conflict-based search with optimal task assignment. In *AAMAS*.

Keskin, M. O.; Cantürk, F.; Eran, C.; and Aydoğan, R. 2024. Decentralized multi-agent path finding framework and strategies based on automated negotiation. *JAAMAS*, 38.

Koyfman, D.; Atzmon, D.; Shperberg, S.; and Felner, A. 2025. Minimizing Fuel in Multi-Agent Pathfinding. In *SoCS*. To appear.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021a. Anytime multi-agent path finding via large neighborhood search. In *IJCAI*.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search. In *AAAI Conference on Artificial Intelligence*, 10256–10265.

Li, J.; Ruml, W.; and Koenig, S. 2021. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. In *AAAI*, 12353–12362.

Li, J.; Surynek, P.; Felner, A.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2019. Multi-Agent Path Finding for Large Agents. In *AAAI*, 7627–7634.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. S.; and Koenig, S. 2021b. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11272–11281.

Maliah, A.; Atzmon, D.; and Felner, A. 2025. Minimizing Makespan with Conflict-Based Search for Optimal Multi-Agent Path Finding. In *AAMAS*, 1418–1426.

Morag, J.; Gabay, N.; Koyfman, D.; and Stern, R. 2025. Should Multi-Agent Path Finding Algorithms Coordinate Target Arrival Times? In *SoCS*. AAAI Press.

Nebel, B. 2020. On the computational complexity of multi-agent pathfinding on directed graphs. In *ICAPS*.

Okumura, K. 2023. LaCAM: Search-based algorithm for quick multi-agent pathfinding. In *AAAI*.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*.

Pertzovsky, A.; Stern, R.; Felner, A.; and Zivan, R. 2025. MCGA: Multi-agent Corridor Generation for Multi-Agent Environments. In *IJCAI*.

Pertzovsky, A.; Stern, R.; and Zivan, R. 2024a. CGA: Corridor Generating Algorithm for Multi-Agent Environments. In *IROS*, 3455–3462.

Pertzovsky, A.; Stern, R.; and Zivan, R. 2024b. CGA: Corridor Generating Algorithm for Multi-Agent Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Pohl, I. 1973. The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving. In *IJCAI*, 12–17.

Ren, Z.; Rathinam, S.; and Choset, H. 2021. MS*: A New Exact Algorithm for Multi-agent Simultaneous Multi-goal Sequencing and Path Finding. In *ICRA*.

Ren, Z.; Rathinam, S.; and Choset, H. 2023. CBSS: A New Approach for Multiagent Combinatorial Path Finding. *IEEE Transactions on Robotics*, 39(4): 2669–2683.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Meta-agent Conflict-based search for optimal multi-agent pathfinding. In *SoCS*.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence Journal (AIJ)*, 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195: 470–495.

Sherma, Y.; Weiss, E.; and Salzman, O. 2025. From Agent Centric to Obstacle Centric Planning: A Makespan-Optimal Algorithm for the Multi-Agent Warehouse Rearrangement Problem. In *SoCS*. AAAI Press.

Shofer, B.; Shani, G.; and Stern, R. 2023. Multi agent path finding under obstacle uncertainty. In *ICAPS*, 402–410.

Silver, D. 2005. Cooperative Pathfinding. In *AIIDE*, 117–122.

Standley, T. 2010a. Finding optimal solutions to cooperative pathfinding problems. In *AAAI conference on artificial intelligence*, 173–178.

Standley, T. 2010b. Finding Optimal Solutions to Cooperative Pathfinding Problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, 173–178. AAAI Press.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. S.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Symposium on Combinatorial Search (SoCS)*.

Surynek, P. 2010. An optimization variant of multi-robot path planning is intractable. In *AAAI Conference on Artificial Intelligence*.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In *ECAI*, 810–818.

Thayer, J. T.; and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In *IJCAI*, 674–679.

Wagner, G.; and Choset, H. 2017. Path planning for multiple agents under uncertainty. In *ICAPS*, 577–585.

Yu, J.; and LaValle, S. 2013a. Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X*, 157–173. Springer.

Yu, J.; and LaValle, S. 2013b. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI Conference on Artificial Intelligence*, volume 27, 1443–1449.

Zhang, H.; Li, J.; Surynek, P.; Kumar, T. K. S.; and Koenig, S. 2022. Multi-agent path finding with mutex propagation. *Artificial Intelligence*, 311: 103766.

Zhang, Y.; Chen, Z.; Harabor, D.; Bodic, P. L.; and Stuckey, P. J. 2024. Planning and Execution in Multi-Agent PathFinding: Models and Algorithms. In *ICAPS*.