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

Giacomo Lodigiani, giacomo.lodigiani@mail.polimi.it

1. Introduction to the research topic

Multi-Agent Path Finding (MAPF) is a problem in the broader field of Multi-Robot Systems in which multiple agents must plan paths to preassigned targets, avoiding collisions. Multi-Agent Pickup and Delivery (MAPD) is an extension of MAPF in which agents have also the freedom to assign themselves to targets which, unlike in the MAPF problem, are not a fixed set but may change at any time step. These problems are mathematically modeled through a formal framework which has seen a growing interest in recent years, due to its great practical relevance especially in the logistics field. However, we will argue that there is still a gap between theory and reliable applications and we will prospect some future work in this direction.

Conferences and Journals
Multi-Agent Path Finding and Multi-Agent Pickup and Delivery belong to the research area of Multi-Robot Systems which, in turn, lies in the intersection of broader research fields like Artificial Intelligence (AI) and Autonomous Robotics. In selecting the most important conferences and journals in these fields, we considered parameters like the H5-index of publications, the Impact Score[1] and the average acceptance rate.
The most relevant conferences with respect to our problem are:

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

The most relevant journals with respect to our problem are:

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

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

1.1. Preliminaries

To better understand the topic and the problem formulations in the next sections, some basic definitions are needed.

Definition 1.1. An *undirected graph* $G = (V, E)$ is a mathematical structure that consists of a set of vertices, or nodes, V and a set of edges $E \subseteq V^2$, which are non ordered pairs of vertices.

Definition 1.2. A vertex v_i is said to be neighbor, or *adjacent*, of a vertex v_j in some graph G if v_i is connected to v_j through an edge.

In the following we will present the classical formulations of the MAPF and MAPD problems [20][10].

1.1.1 The MAPF problem

A MAPF problem instance consists of:

- A given finite connected undirected graph $G = (V, E)$, whose vertices V correspond to locations and whose edges E correspond to connections between locations that the agents can move along.
- A given set of M agents $\{a_i | i \in [M]\}$[2]. Each agent a_i has a start vertex $s_i \in V$ and a goal vertex $g_i \in V$ (that represents the preassigned target). All start vertices are pairwise different. All goal vertices are also pairwise different.

At each discrete time step, each agent a_i either moves to an adjacent vertex or waits at the same vertex. Let $\pi_i(t) \in V$ denote the vertex occupied by agent a_i at time step t.

Definition 1.3. A *vertex collision* is a tuple $\langle a_i, a_j, v, t \rangle$, where agents a_i and a_j occupy the same vertex $v = \pi_i(t) = \pi_j(t)$ at the same time step t.

Definition 1.4. An *edge collision* is a tuple $\langle a_i, a_j, u, v, t \rangle$, where agents a_i and a_j traverse the same edge (u, v), where $u = \pi_i(t) = \pi_j(t+1)$ and $v = \pi_j(t) = \pi_i(t+1)$, in opposite directions between time steps t and $t+1$.

Definition 1.5. A *path* $\pi_i = \langle \pi_i(0), \pi_i(1), ..., \pi_i(T_i), \pi_i(T_i+1), ... \rangle$ for agent a_i satisfies the following conditions:

1. The agent starts at its start vertex, that is, $\pi_i(0) = s_i$.
2. The agent ends at its goal vertex at the *arrival time* T_i, which is the minimal time step T_i such that, for all time steps $t = T_i, ..., \infty$, $\pi_i(t) = g_i$.
3. The agent always either moves to an adjacent vertex or does not move between two consecutive time steps, that is, for all time steps $t = 0, ..., \infty$, $(\pi_i(t), \pi_i(t+1)) \in E$ or $\pi_i(t+1) = \pi_i(t)$.

Definition 1.6. The *makespan* $max_{i \in [M]} T_i$ of a MAPF plan is the maximum of the arrival times of all agents at their goal vertices.

Definition 1.7. The *flowtime* $\sum_{i \in [M]} Ti$ of a MAPF plan is the sum of the arrival times of all agents at their goal vertices.

A MAPF plan consists of a path π_i for each agent a_i. A MAPF solution is a MAPF plan whose paths are collision-free. The problem of MAPF is to find a solution with the smallest makespan or flowtime.
While single agent path finding is tractable (Dijkstra [4]), MAPF is NP-hard to solve optimally for both makespan minimization [21] and flowtime minimization [26]. The optimal makespan and optimal flowtime of any MAPF problem instance are both bounded by $O(|V|^3)$ [27], however, as stated, it is NP-hard to find a solution with the minimum makespan.

[2] We let $[M]$ denote the positive integer set $\{1, ..., M\}$, representing the number of agents.

1.1.2 The MAPD problem

A MAPD problem instance consists of:

- A given finite connected undirected graph $G = (V, E)$, whose vertices V correspond to locations and whose edges E correspond to connections between locations that the agents can move along.
- A given set of M agents $\{a_i | i \in [M]\}$. Each agent has an initial vertex. All initial vertices are pairwise different.
- A task set $\mathcal{T}$ that contains the set of unexecuted tasks in the system. The task set changes dynamically as, at each time step, new tasks can be added to the system. Each task $\tau_j \in \mathcal{T}$ is characterized by a pickup vertex $s_j \in V$ and a delivery vertex $g_j \in V$ and is added to the system at an unknown (finite) time step. A task is known and available for execution only from the time step on when it has been added to the system.

At each discrete time step, each agent a_i either moves to an adjacent vertex or waits at the same vertex. Let $\pi_i(t) \in V$ denote the vertex occupied by agent a_i at time step t. When the system starts (at time step 0), agent a_i starts at its given initial vertex $\pi_i(0)$.

Definition 1.8. A *path* $\pi_i = \langle \pi_i(0), \pi_i(1), ..., \pi_i(T_i), \pi_i(T_i + 1), ... \rangle$ for agent a_i satisfies the following condition: The agent always either moves to an adjacent vertex or does not move, that is, for all time steps $t = 0, ..., \infty$, $(\pi_i(t), \pi_i(t + 1)) \in E$ or $\pi_i(t + 1) = \pi_i(t)$.

Definition 1.9. An agent is called *free* if and only if it is currently not executing any task. Otherwise, it is called *occupied*.

A task can be assigned to one agent at a time. A free agent can be assigned any task $\tau_j \in \mathcal{T}$. In order to execute task τ_j, it then has to move first from its current vertex to the pickup vertex s_j of the task and then from there to the delivery vertex g_j of the task. When the agent reaches the pickup vertex, it starts to execute the task and removes the task from $\mathcal{T}$. When it reaches the delivery vertex, it finishes executing the task, which implies that it becomes free again and is no longer assigned the task. Any free agent can be assigned any task in the task set.

Definition 1.10. The *service time* is the average number of time steps needed to finish executing each task after it was added to the system.

Definition 1.11. The *makespan* is the earliest time step when all tasks are finished.

The problem of MAPD is to find collision-free paths for the agents to finish executing all tasks. Being a generalization of MAPF, also MAPD is NP-hard to solve optimally and the effectiveness of a MAPD algorithm is evaluated by the service time or makespan.

1.2. Research topic

Multi-Agent Path Finding is a type of multi-agent planning problem in which the objective is to plan paths for multiple agents on a graph, where the fundamental constraint is that the agents will be able to follow these paths concurrently without colliding with each other. MAPF has a range of relevant applications which are becoming more and important, including automated warehouses [25], autonomous vehicles, robotics, and videogames. A key characteristic of standard MAPF is that the set of starting and ending vertices corresponding to each task are fixed, and each task is preassigned to an agent. These limitations can be a problem in real world applications, for example in warehouses, where tasks are continuously added to the system and cannot be preassigned to agents in advance; to bridge this gap, more general frameworks have been developed. In the *Lifelong* MAPF formulation [9], new tasks are assigned to different Windowed MAPF (collisions need to be resolved only for the first w time-steps) instances, and subsequent re-plans are used in order to solve the various instances; task assignment is still fixed and considered external to the problem. The more general formulation, named Multi-Agent Pickup and Delivery, combines long-term planning and target assignment and is the more suited to represent a dynamic application environment.

2. Main related works

2.1. Classification of the main related works

Literature in the field of MAPF is broad and rich of theoretical results; on the other hand, research on more articulated (and closer to model real scenarios) problems like MAPD has just started but is growing every year as well as the practical relevance of its applications.
The first dimension which allows us to classify the various works is this particular area of Multi-Agent Systems is the the definition of the type of problem they intend to solve. In the literature it is possible to find slight variations of the same problem under different names; we will consider the subdivision proposed by Ma [10], that is characterized by sharp differences among the various classes.

- **MAPF**: the classical one-shot problem of Multi-Agent Path Finding.
- **Anonymous MAPF**: allows the freedom of assigning targets to agents, combining the problem of path planning and target assignment.
- **TAPF**: Target Assignment and Path Finding, still combines the problem of path planning and target assignment, but agents are divided in teams and an agent can only be assigned a target if that target is given to its team (allocation of targets to teams is predetermined and fixed). It can be considered a generalization of MAPF and Anonymous MAPF.
- **MAPD**: combines path planning and target assignment but, differently from previous problems, is long-term, meaning that new tasks may enter at any time step.

Another direction followed by research is trying to close the gap between the problem formulation and real world applications [22]; some advancements in this area will be presented in Section 2.2. This gap is a limit but also offers great possibilities for new ideas which will ultimately lead to the development of a new generation of more reliable, secure, and efficient Multi-Agent Systems.

2.2. Brief description of the main related works

In this section we will present some details on the various problem formulations and a brief review of algorithms and techniques proposed to adapt those theoretical frameworks to more concrete applications.

2.2.1 Problem formulations

As previously highlighted, while for the classical MAPF problem the theoretical background is consolidated [20], for other variants like MAPD consensus on a common theoretical basis has yet to be reached. For our research, to overcome this issue, we decided to adopt the framework proposed by Ma [10] which, to the present day, is the most structured and coherent. The main aspects of this framework are presented in Section 2.1; more in depth analysis of the MAPF and MAPD problems can be found in section 1.1.1 and 1.1.2.
Another formulation of the long-term problem is given by Li et al. [9], under the name of *Lifelong* MAPF: here the problem of target assignment is not considered, and this allows a decomposition into sequence of Windowed MAPF instances, in which standard MAPF algorithms can be used.
A different aspect of long term operation is considered by Svancara et al. [23] in the definition of *Online* MAPF: in this setting new agents can enter at any time step and task assignment is not necessary since every new agent is associated with a specific start and goal.

2.2.2 Algorithms

Various algorithms exists to solve MAPF problems optimally, in the following we present two of the most notable examples.

- *Conflict-Based Search* (CBS) [17] is a two-level complete and optimal MAPF algorithm that does not convert the problem into a single 'joint agent' model. At the high level, a search is performed on a Conflict Tree (CT) which is a tree based on conflicts between individual agents. Each node in the CT represents a set of constraints on the motion of the agents. At the low level, fast single-agent searches are performed to satisfy the constraints imposed by the high level CT node. In many cases this two-level formulation enables CBS to examine fewer states than a classical algorithm based on A^* search over the joint space of agents [5] while still maintaining optimality.
- *Increasing Cost Tree Search* (ICTS) [18] ICTS is another example of complete and optimal two-level search algorithm. The high-level phase of ICTS searches the increasing cost tree for a set of costs (cost per agent). The low-level phase of ICTS searches for a valid path for every agent that is constrained to have the same cost as given by the high-level phase. A compact data-structure called multi-value decision diagram (MDD) [19] is involved to store all single-agent paths of a certain length, for each agent.

Since MAPF is NP-hard to solve optimally, bounded sub-optimal algorithms were developed to allow applications in time constrained or real-time situations. Some examples are *Bounded Sub-optimal CBS* (BCBS) and *Enhanced CBS* (ECBS) [3], which are derived from standard CBS and exploit *Focal Search*, an approach based on a bounded sub-optimal variant of A^* called $A^*_\in$ [15].
Regarding the MAPD problem, research has started only recently, so fewer algorithms have been proposed. Ma et al. [12] proposed 3 online algorithms divided in two categories: decoupled (where each agent assigns itself to tasks and computes its own collision-free paths given some global information) and centralized.

- *Token Passing* (TP) is a decoupled algorithm based on a token, a synchronized shared block of memory that contains the current paths of all agents, the task set, and the task assignments that record which tasks are currently assigned to which agent. System initializes the token with the trivial paths where all agents rest at their initial vertices. At each time step, the system adds all new tasks, if any, to the task set. Any agent that has reached the end of its path in the token requests the token once per time step. The system then sends the token to each agent that requests it, one after the other. The agent with the token chooses a task from the candidate task set such that no path of other agents in the token ends at the pickup or delivery vertex of the task, because it can then find a path to the pickup vertex and then a path to the delivery vertex of such a task and safely rest at the delivery vertex. Finally, the agent returns the token to the system and moves along its path in the token (if it can't find a feasible task it rests where it is sure not to cause deadlocks).
- *Token Passing with Task Swaps* (TPTS) is a decoupled algorithm similar to TP except that its task set now contains all unexecuted tasks, rather than only unassigned tasks. This means that an agent with the token can assign itself not only a task that is not assigned to any agent but also a task that is already assigned to another agent as long as that agent is still moving to the pickup vertex of the task. This might be beneficial when the former agent can move to the pickup vertex of the task in fewer time steps than the latter agent. The latter agent is then no longer assigned the task and no longer needs to execute it. The former agent thus sends the token to the latter agent so that the latter agent can try to assign itself a new task.
- *Central* is a centralized algorithm that makes decisions for multiple agents at a time. Similar to TPTS, *Central* allows agents that have just become free to consider not only unassigned tasks but also all unexecuted tasks, including the ones that have been assigned to agents, in the task set. Unlike TPTS, *Central* uses a centralized target-assignment algorithm, the Hungarian method [8], to assign (or reassign) tasks to agents and allows all free agents to consider tasks that have just been added to the system. It uses the centralized MAPF algorithm CBS to plan paths for multiple agents.

As seen in experimental results [12], TP is the only algorithm that can be used for real-time lifelong operation, at the cost of increased makespan and service time.

2.2.3 Robustness

The term robustness describes the property of a MAPF or MAPD algorithm of being able to complete all the planned tasks even in case some unexpected event, not considered in the original formulation of the problem but possible in real applications, forces some deviation of the execution from the original plan.
In the MAPF setting, Atzmon et al. [1] defined as *k-robust* a plan that does not have any *k*-delay conflicts. Informally, this means that no conflicts will occur even if some of the agents are delayed by up to *k* time steps; the authors proposed an algorithm to solve this problem optimally based on CBS, called *k-Robust CBS*. Starting from the idea of *k*-robustness, Atzmon et al. [2] defined as *p-robust* a solution that is conflict-free with probability at least p, even though unexpected delays may occur. This notion is, arguably, a more realistic form of robustness, since it does not assume a strict limit on the number of unpredictable delays per agent.
For MAPD algorithms key elements are environmental characteristics, namely well-formedness, that can provide a sufficient condition to allow for long-term robustness [10]. The intuition is that agents should only be allowed to rest (that is, stay for a long period without an intention to move away) at vertices, called endpoints, where they cannot block other agents. According to Ma et al. [12], a MAPD problem instance is well-formed if and only if:

1. The number of tasks is finite.
2. There are no fewer non-task endpoints (designated vertices that allow the agents to rest) than the number of agents.
3. For any two endpoints, there exists a path between them that traverses no other endpoints.

So, until now, long-term robustness in the MAPD framework has been studied only inside the original theoretical setting of the problem; long-term robustness in real application settings, where delays, failures and adversarial events may occur, still remains an open question.

2.2.4 Kinematic constraints

So far we have considered agents as abstract entities; in reality they are often robots and, as such, they are subject to physical constraints like minimum turning radius, maximum velocity, maximum acceleration, etc.; to introduce these physical limitations in the formulation of the problem, some solutions have been proposed.
Hoenig et. al. [7] presented MAPF-POST, an approach that makes use of a simple temporal network to post-process a MAPF plan in polynomial time to create a plan-execution schedule that works on non-holonomic (not all degrees of freedom can be controlled at the same time) robots, takes their maximum translational and rotational velocities into account, provides a guaranteed safety distance between them, and exploits slack (defined as the difference of the latest and earliest entry times of locations) to absorb imperfect plan executions and avoid time intensive replanning in many cases. A similar approach was later followed by Hoenig et al. [6]: their solution exploits a particular type of graph, called Action Dependency Graph, that captures the action precedence relationships of a MAPF solution and can be used to enforce these relationships on real robots with higher-order dynamics. More recently, Ma et al. [11] proposed, for the MAPD problem, an improved version of TP, called TP-SIPPwRT, keeping into account cinematic constraints. TP is made more effective using a novel combinatorial search algorithm, called Safe Interval Path Planning with Reservation Table (SIPPwRT) for single-agent path planning. SIPPwRT uses an advanced data structure that allows for fast updates and lookups of the current paths of all agents in an online setting. The resulting MAPD algorithm TP-SIPPwRT takes kinematic constraints of real robots into account directly during planning, computes continuous agent movements with given velocities that work on non-holonomic robots rather than discrete agent movements with uniform velocity, and is complete for well-formed MAPD instances.

2.2.5 Application environment

Since MAPF and MAPD have great potential in logistics applications, it is no surprise that the warehouse environment is often used as benchmark for algorithms implementations [20].

Li et al.[9], inspired by the dynamic nature of a warehouse, defined the problem of *Lifelong* MAPF (see Section 2.2.1) and were among the first to underline the importance of long-term execution for reliable applications in such environment.
Salzman and Stern [16] proposed a different point of view in their analysis of the open challenges for future research: they suggested to find ways to optimize the warehouse layout in order to get, in average, better solution to the task assignment and planning problem.
Another logistics problem, airport surface operations, is considered by Morris et al. [14]; this problem consists of a set of complex logistics tasks that nowadays involve coordination of humans and machines. The authors propose a fully autonomous approach, through MAPF and kinematic constraints implemented via a simple temporal network, which could lead to safer operations and to a more efficient use of existing surface area to meet increasing demand.
Other interesting applications of MAPF and MAPD outside the logistics domain are service robots [24] and videogames [13].

2.3. Discussion

In the near future, Autonomous Robotics and Artificial Intelligence are going to be an integral part of everyday life in a lot of different sectors like transportation, logistics, search and rescue, healthcare. For this revolution to be successful, two prerequisites are crucial: a strong theoretical foundation, needed to ensure coherent and efficient solutions, and a solid implementation experience in order to guarantee the robustness of those solutions and their capability to overcome the most common problems of autonomous operation.
We have analyzed these two dimensions in the sub-field of Multi-Agent Systems and we have underlined that, while theory is relatively mature, still a lot of work has to be done towards implementations. MAPD is a promising attempt trying to bring theory one step closer; however still nothing can be said on whether its long term guarantees will hold in a context where execution is not perfect and the environment could be, in some cases, adversarial.

References

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

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

[3] Barer, M., Sharon, G., Stern, R., and Felner, A. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS)* (2014).

[4] Dijkstra, E. W. A note on two problems in connection with graphs. *Numerische mathematik 1, 1* (1959), 269–271.

[5] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4, 2* (1968), 100–107.

[6] Hoenig, W., Kiesel, S., Durham, A. T. J., and Ayanian, N. Persistent and robust execution of mapf schedules in warehouses. *IEEE Robotics and Automation Letters, 4, (2)* (2019), 1125–1131.

[7] Hoenig, W., Kumar, S., Cohen, L., Ma, H., amd N. Ayanian, H. X., and Koenig, S. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)* (2016), pp. 477–485.

[8] Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2* (1955), 83–97.

[9] Li, J., Tinka, A., Kiesel, S., Durham, J., Kumar, S., and Koenig, S. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2020).

[10] Ma, H. Target assignment and path planning for navigation tasks with teams of agents. *PhD thesis, Department of Computer Science, University of Southern California, Los Angeles (California)* (2020).

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

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

[13] Ma, H., Yang, J., Cohen, L., Kumar, T. K. S., and Koenig, S. Feasibility study: Moving non-homogeneous teams in congested video game environments. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment* (2017), pp. 270–272.

[14] Morris, R., Pasareanu, C. S., Luckow, K., Malik, W., Ma, H., Kumar, T. K. S., and Koenig, S. Planning, scheduling and monitoring for airport surface operations. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2016).

[15] Pearl, J., and Kim, J. H. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence 4* (1982), 392–400.

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

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

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

[19] Srinivasan, A., Ham, T., Malik, S., and Brayton, R. K. Algorithms for discrete function manipulation. In *Proceedings of the IEEE International Conference on Computer-aided Design. Digest of Technical Papers* (1990), pp. 92–95.

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

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

[22] Svancara, J. Bringing multi-agent path finding closer to reality. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2018), pp. 1784–1785.

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

[24] Veloso, M., Biswas, J., Coltin, B., and Rosenthal, S. Cobots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the International Conference on Artificial Intelligence* (2015), pp. 4423–4429.

[25] Wurman, P. R., D'Andrea, R., and Mountz, M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine 29, 1* (2008).

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

[27] Yu, J., and Rus, D. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *Algorithmic Foundations of Robotics XI, Springer Tracts in Advanced Robotics, Vol. 107* (2015), pp. 729–746.