IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 April 2024, accepted 18 April 2024, date of publication 22 April 2024, date of current version 29 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392305

SURVEY

A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding

JEAN-MARC ALKAZZI[1,2] AND KEISUKE OKUMURA[3,4], (Member, IEEE)
[1]IDEALworks GmbH, 80992 Munich, Germany
[2]FEMTO-ST Institute, UMR 6174 CNRS, University Bourgogne Franche-Comté, 25000 Belfort, France
[3]Department of Computer Science and Technology, University of Cambridge, CB3 0FD Cambridge, U.K.
[4]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan

Corresponding author: Jean-Marc Alkazzi (jean-marc.alkazzi@idealworks.com)

This work was supported in part by IDEALworks GmbH and in part by JST ACT-X under Grant JPMJAX22A1.

ABSTRACT This review paper provides an in-depth analysis of the latest advancements in applying Machine Learning (ML) to solve the Multi-Agent Path Finding (MAPF) problem. The MAPF problem is about finding collision-free paths for multiple agents to travel from their source to goal locations in a known environment. This method underpins a range of advanced, large-scale automated systems, notably in warehouse logistics. The existing research on conventional MAPF is extensive; however, recent developments in ML have notably augmented the capabilities of MAPF techniques. This research seeks to thoroughly investigate the emerging field focused on using ML to help solve the MAPF problem. It aims to highlight the transformative potential of ML in enhancing the efficiency and effectiveness of multi-agent systems in navigating and coordinating in complex environments. Our study comprehensively examines the entire MAPF process, encompassing environment representation, path planning, and solution execution.

INDEX TERMS Machine learning, multi-agent pathfinding, multi-robot system.

I. INTRODUCTION

Navigating a team of agents to their respective destinations is a crucial technology in the rapidly evolving era of automation. In fact, a wide range of systems can benefit from multi-agent navigation technologies. A prime example is warehouse automation using automated guided vehicles [1], [2], where multiple robots are tasked with transporting packages. Here, *coordination* among robots is crucial to increase the system's efficiency and avoid *collisions* that could lead to cascading crashes or *deadlocks* that impair system throughput. This fundamental problem manifests in various scenarios, such as autonomous intersection management for self-driving cars [3], railway scheduling for multiple trains [4], [5], generating realistic motions for animated agents in video games [6], planning airport surface operations involving diverse mobile entities [7], automated vehicle parking [8], swarm drone operation [9], to mention just a few.

The field of multi-agent navigation poses intricate challenges in artificial intelligence and robotics because, beyond the technical challenges encountered in single-agent navigation, it introduces unique difficulties in orchestrating coordination among multiple agents. These include design choices of centralized versus decentralized control schemes, dealing with entity heterogeneity, optimizing communication methods, and determining the extent of shared knowledge among agents. Among these multi-agent system-specific challenges, an indispensable component is designing coordinated movements for mobile agents. This aspect is fundamentally encapsulated in the *multi-agent path finding (MAPF)* problem [10], which is the focus of this paper.

A. MAPF DEFINITION AND BASIS

MAPF is a *planning* problem that aims to find collision-free paths for multiple agents within a known environment, enabling their transition from their respective start to goal locations. The standard MAPF formulation includes a graph representing the agents' workspace and a set of start-goal vertices. Agents are assumed to move synchronously, each time step allowing them to remain idle or move to an adjacent vertex. A solution to this problem is a collection

The associate editor coordinating the review of this manuscript and approving it for publication was Yifan Zhou.

of collision-free paths, each corresponding to an individual agent, ensuring that no two agents simultaneously occupy the same vertex or edge. Beyond finding feasible solutions, the goal is generally to identify ''good'' solutions, characterized by minimizing a solution cost. This cost could be the *makespan* (i.e., the duration of the longest path among all agents), the flowtime (i.e., the total sum of all agents' path costs; aka. sum-of-costs), or the sum of energy consumption for robotic motions. Theoretical analyses have demonstrated that solving the MAPF problem optimally with various solution metrics is NP-hard [11], [12], [13]. This complexity persists even when limiting the scope to planner graphs [14] or grid structures [15], [16], and also in seeking bounded suboptimal solutions within a small constant factor [17]. These findings underscore the computational challenges in finding near-optimal MAPF solutions, particularly when involving a large number of agents.

B. CUTTING-EDGE MAPF STUDIES

Despite the theoretical complexities of finding (near-)optimal solutions, recent dramatic advances in MAPF algorithms have made it possible to solve large MAPF problems. Nowadays, optimal solutions can be found in less than a minute for systems with tens of agents [18], [19], [20], [21], [22], and plausible suboptimal solutions for hundreds of agents in mere seconds [23], [24], [25], [26]. Remarkably, the latest work [27], [28], [29] can manage thousands of agents within a few seconds on consumer-grade laptops. These advancements have not only improved planning capabilities but also expanded the research scope of MAPF in related technological fields. This includes going beyond the simplified assumption of uniform travel time between vertices [30], [31], [32], extending MAPF to various robotic systems like manipulators [33], [34], [35], moving beyond the typical one-shot planning problems in MAPF [2], [36], [37], joint planning with target assignments [38], [39], [40], [41], and executing MAPF solutions with actual robots under timing uncertainties [42], [43], [44], [45], among others. This evolution broadens the horizons of MAPF technologies, making them increasingly realistic and practical for implementing efficient multi-agent navigation.

C. MAPF DEPLOYMENT CHALLENGES

Although the remarkable progress of MAPF technologies has been made, their deployment in real-world scenarios still encounters significant challenges. Even from a basic planning perspective, achieving real-time, scalable, and near-optimal solutions remains an intractable task [46], [47], detrimental to major applications. Furthermore, real-world conditions introduce complex issues, such as kinodynamic constraints that challenge conventional MAPF methods, unstable network environments that rule out centralized controls, which most MAPF methods assume, and unpredictable human interactions in the operational process. Robust and resilient methodologies are also essential to ensure that a single robot's failure does not trigger a cascade of malfunctions, potentially halting the entire system's operation [48], [49]. Consequently, *there remains a considerable gap between cutting-edge lab-scale studies on MAPF and their deployments in real-world applications*.

D. POTENTIAL FOR MACHINE LEARNING

On a parallel front, *machine learning (ML)*, especially *deep learning*, has demonstrated outstanding success in addressing complex challenges across various fields [50]. Notable achievements include the development of superhuman-level AIs for games [51], [52], [53], breakthroughs in protein structure prediction, a fundamental issue in biology [54], the generation of realistic images [55], advancements in text generation exemplified by large language models [56], [57], and significant progress in robotics [58], [59]. These accomplishments underscore the profound and transformative impact of ML in solving intricate real-world problems.

This success leads us to envision that *the proper incorporation of ML techniques could significantly enhance the development of more practical, scalable, and robust MAPF technologies*, bridging the gap between lab-scale research and real-world applications. In fact, various studies integrating ML and MAPF have emerged in recent years. These include augmenting classical MAPF solvers with ML [60], [61], [62], [63], algorithm selection as an ML prediction task to identify suitable MAPF algorithms for each situation [64], [65], [66], developing decentralized and adaptive agent policies through learning [67], [68], [69], and enhancing graph generation through generative modeling to aid MAPF algorithms in deriving superior solutions [70], [71], among others.

E. PAPER OBJECTIVES

To this end, this survey paper aims to systematically organize the emerging technologies in the field of *MAPF enhanced by ML*, emphasizing the synergy between them. Our objective extends beyond simply documenting the integration of classical MAPF planning with ML. Instead, we aim to comprehensively cover relevant studies in this domain, aligning with the motivations observed thus far. Accordingly, this paper categorizes existing studies based on three perspectives as follows:

- **Representation**, modeling and designing workspaces and MAPF problem statements, for subsequent processing.
- **Planning**, synthesizing a sequence of actions for agents to execute.
- **Execution**, concerning the implementation of these actions by agents, considering the various uncertainties encountered in real-world environments.

Figure 1 offers a visual illustration, elucidating the intuitive interconnections among these three aspects. By adopting this framework, the paper presents a unified perspective on how ML can augment MAPF and its associated technologies, thereby shedding light on potential future integrations and innovations.

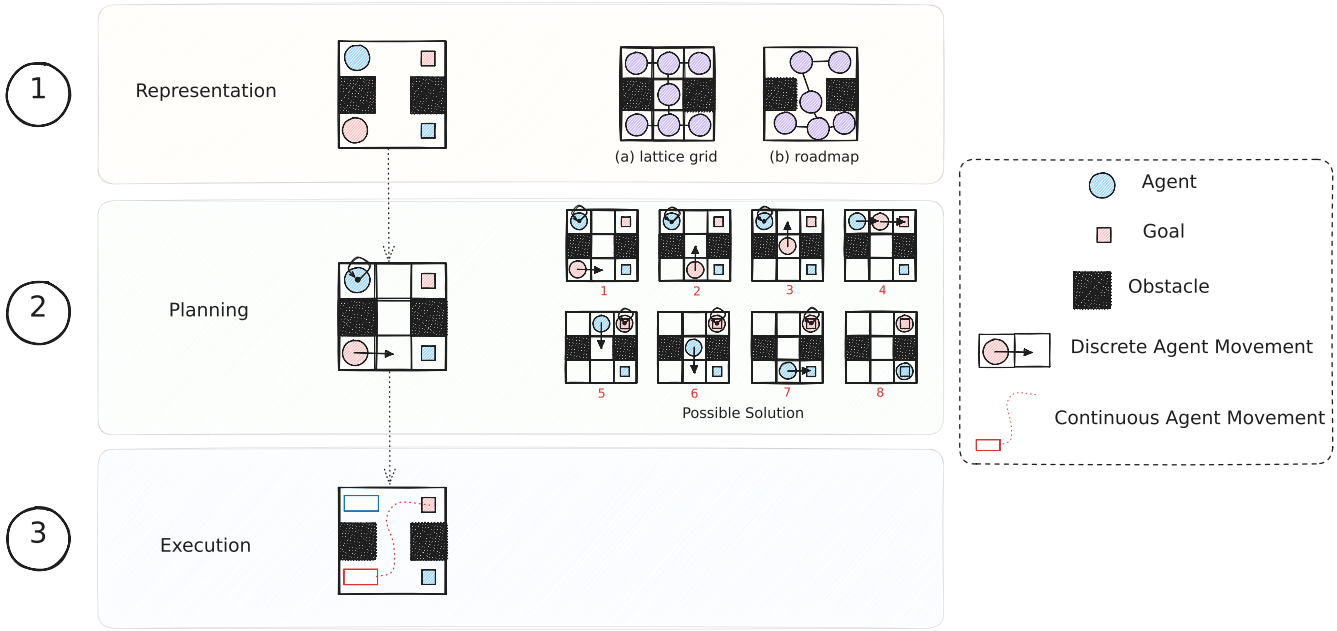


FIGURE 1. MAPF stages. Using a discrete (1) representation such as a grid or a roadmap generated by sampling-based motion planning methods, MAPF algorithms perform (2) planning that finds a sequence of collision-free actions for each agent. The resulting paths are (3) executed by agents grounded in the real world.

This paper stands out as the first comprehensive survey that broadly explores how ML techniques can improve MAPF-related challenges. While existing surveys on MAPF [72], [73], [74], [75], provide valuable insights, they primarily focus on methods without ML and thus differ from our scope. Recent surveys [76], [77], [78], [79], [80] focus more on the role of ML in planning for MAPF, especially in designing agent policies that can replace centralized MAPF methods. However, their scope is limited to this specific planning perspective. In contrast, this review provides unique insights into integrating MAPF and ML across all three dimensions mentioned above, thus providing a holistic view of the field.

F. OUT OF SCOPE TOPICS

Since multi-agent navigation has a high impact on industrial applications, there are many research areas not limited to MAPF that aim to implement multi-agent navigation. Among others, to avoid an explosion of discussion content, this paper does *not* cover so-called *reactive* approaches, where agents locally avoid collisions in continuous spaces. Typical methods of this class include velocity obstacles [81], [82], [83], artificial potential field [84], [85], social force [86], dynamic window approach [87], and control barrier function [88], [89]. In general, MAPF-based methods are called *deliberative* approaches, and they have advantages over reactive ones in terms of theoretical guarantees for global agent coordination, such as guarantees of collision-free, deadlock-free, and optimality of agent motions. Meanwhile, reactive approaches often excel at scalability for the number of agents and real-time nature, allowing a team of agents to quickly adapt to dynamic environmental changes.

Target/task assignment [90], deciding where agents should go, which is a domain to which conventional MAPF is closely related as seen in [38], [39], [40], [41], is another topic not covered in this paper. This decision is primarily due to our focus on path planning systems themselves, and partially due to the lack of literature on how ML can improve the joint problems of target assignment and pathfinding.

G. PAPER ORGANIZATION

This paper is structured to first delve into representation, followed by detailed discussions on planning and execution strategies and concluding remarks. In addition to providing a review of the literature, each section includes open questions worthy of future investigation. It is noteworthy that the sections on representation and execution encompass a less substantial number of studies compared to the planning section. This distribution is a deliberate reflection of the current research landscape, and does not diminish the thoroughness of our coverage across the entire field.

H. TERMINOLOGIES

In the context of this paper, the terms *agents* and *robots* are used interchangeably without any notable distinction. The term *environment* is defined as the space in which the agents operate. A *MAPF instance* comprises a graph, a collection of agents, and their respective start-destination pairings. A *MAPF problem* is a typical planning challenge that aims to identify a set of collision-free paths (referred to as a *solution*) for a given MAPF instance. A *MAPF algorithm* or *solver* describes a method employed to address and solve the MAPF problem. *MAPF challenges* encompass the MAPF

problem and associated concerns, such as environment representation and the implementation of solutions given additional constraints.

II. REPRESENTATION

This section addresses how to *represent* MAPF problems to implement efficient multi-agent navigation. Specifically, we discuss three non-trivial perspectives and how ML techniques can be applied to enhance each:

- **Representation for Planning** (Section II-A) focuses on how to generate a suitable graph representation that assists MAPF algorithms in deriving plausible paths while sustaining the computational effort.
- **Environment Optimization** (Section II-B) focuses on adjusting the layout of movable obstacles or other artifacts such as shelves in the environment. This strategic manipulation aims to enhance the effectiveness of MAPF algorithms.
- **Representation for Selection** (Section II-C) addresses how to represent a MAPF instance in order to select the most suitable MAPF algorithm, considering the diverse types of algorithms that exist.

A. REPRESENTATION FOR PLANNING

The initial step in addressing MAPF problems is to answer the following question: *"Where can agents move within the environment?"* Since MAPF is fundamentally a combinatorial search problem in a discretized space, it necessitates representing the agents' environment with a graph $G = (V, E)$, where V represents the set of vertices and E the edges. This representation is crucial as it must accurately reflect navigable areas, forming a foundation for developing collision-free paths. More specifically, this task involves effectively identifying obstacle-free spaces, denoted as C_{free}, within the *configuration space*. The configuration space C is defined as the domain encompassing all potential states of an agent. Subsequently, a graph G, often referred to as *roadmap*, is constructed within C_{free} to facilitate the generation of obstacle-free paths.

Figure 1 includes two commonly used representations for MAPF planning: *(a)* a lattice grid and *(b)* a roadmap constructed by sampling-based motion planning (SBMP) methods [91], such as the Probabilistic Roadmap (PRM) [92]. The lattice grid is simple to construct, while the latter roadmap typically follows three steps: *(i) Vertex sampling* samples vertices V from the agent's collision-free configuration space C_{free}, given the known environment; *(ii) Edge creation* connects sampled vertices with collision-free edges E, and; *(iii) Graph refinement* post-processes the generated graph $G = (V, E)$ to adapt agent or environment-specific constraints and potential optimizations in vertex location, edge cost, and direction. Both representations of grid and roadmaps with SBMP methods establish a navigation graph structure. This ensures agents remain collision-free from static obstacles when positioned at a vertex and while transitioning to another vertex via an edge.

The main challenge in representation for planning is finding a balance between the environment's graph density and the availability of a viable solution, as illustrated in Figure 2. On one hand, a very dense representation may encompass the solution but slow down the planning due to an expanded search space, impacting subsequent planning processes, including inter-agent collision management. On the other hand, a sparser representation might expedite the solving process but at the risk of lacking a valid solution. In fact, the typical representations mentioned above struggle with this challenging trade-off. Grid representations that are too sparse risk having unreachable areas, while those that are too dense require significant planning effort. In contrast, sparse roadmaps with SBMP often overlook the specific structural layout of the environment. For instance, SBMP may result in varying sampling densities in corridors or fail to add vertices in essential areas for agent navigation, but denser roadmaps incur significant planning effort. Since this challenge also occurs in single-agent navigation, researchers have studied how to densify the representation of the environment for single-agent navigation [93], [94], [95]. Multi-agent scenarios pose more complex challenges due to interagent interactions.

In the following, we first explore non-ML approaches to address this representation challenge and then examine how ML can enhance this aspect.

1) NON-ML APPROACHES

The first attempt in this specific topic that we are aware of was made by Henkel and Toussaint [96]. The work proposed to optimize the generation of PRMs for MAPF using stochastic gradient descent [97] and Adam [98] as the optimizer. This strategy proceeds by continuously iterating through the vertex sampling and edge creation steps while minimizing a custom cost function that aims for sparser graphs, leaving enough space between the edges while optimizing travel costs for planning the MAPF solution.

Gao et al. [99] studies a graph embedding method in complex environments, where the resulting graph is used in subsequent planning. The study focuses on increasing free space coverage by defining ad-hoc re-meshing operators, along with mathematical optimization of the geometric relationship between vertices. The approach converts feasibility conditions into differentiable geometric constraints, enabling the mesh optimizer to find feasible solutions through constrained numerical optimization while maximizing metrics for denser robot packing.

Okumura et al. [34] takes a holistic view of solving multi-agent navigation, which transfers to diverse environments and agent kinematics, such as ground robots of different sizes and robotic arms in 3D space. In essence, the work proposed an algorithm combining the gradual development of roadmaps and motion planning for multiple agents, aiming to balance the representation issue in Figure 2.

All these studies [34], [96], [99] leverage optimization techniques without any learning process involved; they

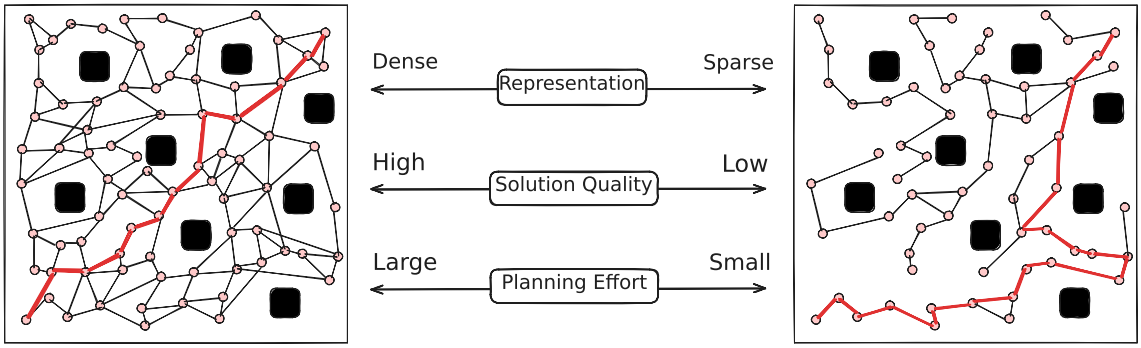


FIGURE 2. Representation trade-offs. Navigation graphs (i.e., roadmaps) are represented by small dots and thin lines, while environments have multiple obstacles represented by black squares. The left roadmap is a 'dense' representation of the workspace, which provides sufficient coverage of the environment and allows the solver to derive plausible solutions, but potentially requires significant planning effort. In contrast, the right presents a 'sparse' one with a smaller search space, having a risk of deriving bad solutions or failure of the planning.

showcase the effectiveness of environment representation in MAPF techniques and inspire the advancement of more sophisticated methods, detailed in the following section.

2) ML APPROACHES

The literature on this topic is still sparse; this part provides a detailed review of two significant studies. The essence of these works is the extension of roadmaps with SBMP to sample sparse yet promising locations for subsequent planning. These regions are identified through offline compute-intensive methods and leveraged in online graph generation, which can be cast as ML prediction problems.

We first examine the construction of Avoidance Critical PRM (ACPRM) [70], which enhances traditional PRM by learning to identify critical areas in the environment for better obstacle avoidance. It defines 'avoidance criticality' as the likelihood of an agent pausing due to dynamic obstacles, with higher criticality in more challenging areas like narrow aisles. ACPRM's self-supervised learning method classifies areas based on their avoidance criticality. It significantly accelerates the search and planning process in multi-agent scenarios up to five orders of magnitude over uniform grid sampling.

Another interesting approach is the construction of Cooperative Timed Roadmaps (CTRMs) [71], which aims to streamline planning by balancing graph sparsity and high-quality solutions. This two-step process involves a Conditional Variational Autoencoder (CVAE) [100], [101] learning to predict agents' next promising locations during offline training. During inference, a combination of the CVAE and a random walk model constructs CTRMs for MAPF. The CVAE leverages previous solutions to predict agents' movements, considering other agents' presence. The random walk model, integrated during inference, enhances CTRMs' adaptability beyond the initial training data.

Open Question 1: What can be beneficial criteria and reliable benchmarks for assessing the quality of environment representation?

A good environment representation has varying criteria, as each study uses different optimization metrics, such as maximizing the coverage of the workspace [99] or minimizing planning effort [71]. Moreover, additional constraints like "move parallel to walls" [96] may be introduced. Identifying and formulating useful metrics is valuable in itself. It is also beneficial for the research community to have a benchmark to assess the qualification of environmental representation generation methods, which contain challenging instances from real-world applications such as narrow aisles and logistics environments with highways.

B. ENVIRONMENT OPTIMIZATION

In typical MAPF studies, environments are often modeled with static layouts that have been predetermined. However, real-world applications frequently pose a different challenge: *the design of effective environment layouts for multi-agent navigation*. The significance of these layouts is paramount as they include various elements that define navigation graphs, such as obstacle locations, shelf positions, and pickup-delivery points in package delivery tasks, which are sometimes modifiable. A carefully designed layout significantly enhances the performance and efficiency of MAPF solvers by complementing their capabilities. This leads to the crucial task of optimizing logistics facility layouts in conjunction with MAPF solvers.

The concept of environment optimization considers the facility layout as a key component in system-wide optimization. It entails adjusting the layout to optimize the performance of selected MAPF solvers. This optimization can be implemented in two ways [102], [103]: *offline*, involving a one-time optimization followed by the deployment of a MAPF solver, and *online*, which includes the dynamic rearrangement of obstacles as agents move, aiming to minimize specific costs, such as flowtime. This dual strategy for layout optimization is illustrated in Figure 3.

Traditionally, layout design in this context has been a manual process conducted by human logistics experts. While

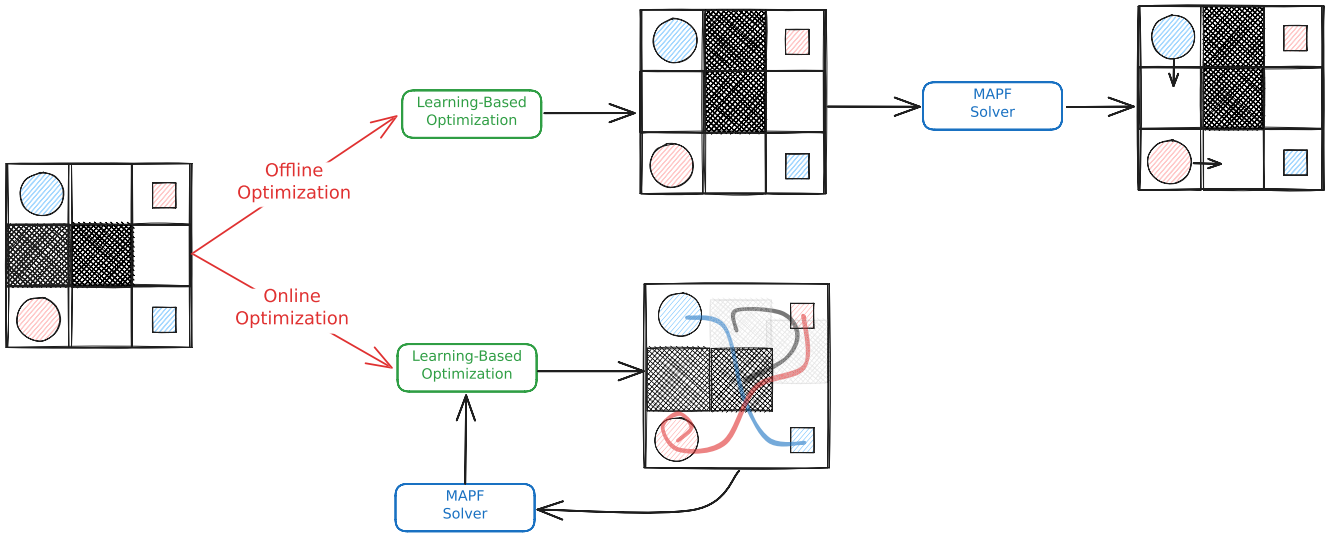


FIGURE 3. Environment layout optimization following the formulation in [102]. Black cells represent movable obstacles. Offline optimization adjusts the layout before applying the MAPF algorithms, while online optimization continuously changes the layout as agents move.

this method can be effective, it may introduce subtle biases that are hard to detect or recognize and could compromise system performance. An alternative approach involves using heuristic search methods to identify effective environment layouts. Indeed, the joint optimization of environments and MAPF has been studied [104], [105], where agents maneuver obstacles to reduce bottlenecks and create shorter paths. However, such search-based strategies are limited to smaller environments, as larger ones lead to combinatorial explosions. This limitation underscores the potential of ML in optimizing such environments.

Several studies have demonstrated the potential of ML in enhancing the optimization of environments. Gao and Prorok [102], [103] introduce learning-based methodologies to optimize obstacle layouts. These methodologies aim to improve the performance of multi-agent navigation in both offline and online scenarios mentioned above. More specifically, these methods use model-free reinforcement learning to adjust the locations of obstacles, which involve encoding environments with deep learning methods such as convolutional neural networks or graph neural networks.

Another example of environment optimization with ML is the work by Zhang et al. [106], [107]. Their study focuses on optimizing warehouse layouts, particularly for offline scenarios and lifelong MAPF, where agents' goals are continuously updated. The environments are represented as grids to assign specific functions, like shelf locations, to each grid cell. To efficiently generate these environment layouts, the initial study [106] combines a deep surrogate model with an evolutionary algorithm that relaxes the computational burden of the optimization process. The follow-up study [107] uses neural cellular automata [108] to create layouts of varying sizes.

Open Question 2: What are efficient transition mechanisms between offline and online environment optimization?

While offline optimization benefits from more computation power and available processing time, it lacks reactivity to real-time environmental changes. On the other hand, online optimization has access to such information under time constraints to generate more efficient layouts. Offline approaches that consider the possibility of failures or changes for more efficient online optimization will be beneficial to real-world scenarios. Similar techniques were studied for the planning phase of the MAPF problem [45], [109].

C. REPRESENTATION FOR SELECTION

Given a set of algorithms called *algorithm portfolio* and a representation of the problem instance, an *algorithm selection* problem is about choosing the best-suited algorithm for the specific input instance [110]. The definition of "best suited" depends on the metric we aim to optimize. Selecting an appropriate algorithm becomes crucial in scenarios where distinct algorithms exhibit varying strengths and weaknesses contingent upon the specific problem instance. The strategic choice of an algorithm, tailored to the unique characteristics of each instance, can significantly enhance the efficiency and effectiveness of the problem-solving process.

ML techniques can benefit algorithm selection due to their feature extraction capabilities that help automate the selection process [111]. This can help uncover relevant features previously unknown and avoid the bias introduced by hand-crafted feature selection. Furthermore, ML-powered algorithm selection simplifies the addition of more algorithms to the portfolio with additional training/fine-tuning, which helps create a portfolio with complementary strengths.

This paradigm, ML-powered algorithm selection, proves particularly advantageous for MAPF because:

- A vast array of MAPF algorithms exists, ranging from optimal approaches [18], [19], [20], [21], [22] to unbounded suboptimal strategies [23], [24], [25],

[26], [27], [28]. This diversity enables the creation of a comprehensive algorithm portfolio.
- No universally dominant MAPF algorithm has been identified that performs optimally across all domains thus far [46]; hence, we must carefully select a suitable algorithm for each instance.
- Additionally, several algorithms are associated with *hyperparameters*, making the case-by-case manual tuning of appropriate parameters challenging.

Thus, we discuss the following ML challenge: *Which solver is best suited for the particular MAPF instance at hand?*

We put this as a representation problem because *algorithm selection entails a significant challenge in representing MAPF instances for ML input*. For instance, some studies represent MAPF instances as images [64], [65], [66], [112], while others utilize manual features [112]. In the following, we first describe a general view of how MAPF algorithm selection happens with different approaches taken in the literature, followed by open questions.

1) METHODOLOGY REVIEW

As illustrated in Figure 4, algorithm selection for MAPF starts with representing a MAPF instance and then trains ML functions such as neural networks to predict the best MAPF algorithm. The algorithm portfolio is typically constructed from optimal MAPF algorithms, such as [22], [113], [114], and [115]. Once trained, such a function can be used as an instant oracle, providing which MAPF algorithms to use depending on the input instance. Typical MAPF situations aim to minimize solution costs, such as the makespan (i.e., the maximum travel time) or the flowtime (i.e., the sum of travel time), while ensuring a solution within given time constraints. Therefore, the best algorithm to be chosen is the one that exhibits the minimum metric while also achieving the fastest solving speed. In specific situations where only optimal algorithms are available [64], [65], [66], [112] depicting the same solution quality, the selector is therefore tasked to choose the fastest algorithm for the given instance.

Sigurdson et al. [64] conducted the first study on this topic, which represents the MAPF instance as an image of the occupancy grid with green and red dots representing agents' start and goal location. The work uses AlexNet [116], a pioneering neural network architecture for image classification, to classify an input as "best solvable using X algorithm from our portfolio."

Several studies then enrich the MAPF algorithm selection. For instance, Kaduri et al. [112] proposed two algorithm selectors. One uses the exact representation from [64] by replacing AlexNet with a modified VGG-16 architecture [117], another prominent image classification model. The other selector relies on manually selected features and an XGBoost model for the prediction. The study done by Ren et al. [65] based itself on the representation in [64], hence using the image representation, while overlaying the single-agent shortest paths for each pair of start-goal. Alkazzi et al. [66] later developed more efficient and simpler neural network designs, using the same representation technique as referenced in [65].

2) CHALLENGES

Input instances can be of varying shapes with different numbers of agents, obstacle density, and possible maximum makespan. As a result, effectively representing MAPF instances for ML models poses non-trivial challenges.

For instance, the issues for image representation stem from attempting to encapsulate spatial and temporal features on a 2D plane, thereby constraining the temporal attribute's value. Additionally, distinguishing between different agents becomes challenging, aggravating the path overlap issue. Figure 5, which is taken from [65] and [66], presents a concrete example; we can see how the density of agents plays a negative role in the clarity of the representation. This, in turn, hinders the performance of the deep learning model trying to extract relevant features for the selection process. The more overlap there is for the single-agent paths, the lower the quality of the input, hence less relevant features and lower selection performance.

Open Question 3: What is the appropriate input instance representation for algorithm selection?
Manual features may bias the learning process [112] while current 2D image representations do not scale to instances with a higher density of agents [65], [66] and are insufficient to relay temporal information. A viable approach might entail employing a 3D representation or assigning a unique color to each agent. In a 3D setup, each slice of the input represents the spatial configuration of the problem at a specific temporal checkpoint, which helps differentiate between path intersection and actual agent collision. Meanwhile, this may require more advanced model architectures that act on sparse input.

Open Question 4: What is the appropriate representation for MAPF on non-grid worlds?
Previously presented approaches have focused on a 2D representation for grid-world environments [64], [65], [66]. This keeps non-grid worlds as an untackled open challenge.

III. PLANNING

The planning phase in MAPF is designed to generate collision-free paths for each agent to reach its goal. These paths are represented as sequences of vertices and edges within a graph, spanning obstacle-free spaces (i.e., C_{free}), as prepared in the previous section. The essence of MAPF lies in deriving such solution paths, making this domain's literature extensive and rich. However, two primary challenges arise.

The primary challenge lies in the fact that, even with considerable progress in MAPF algorithms, achieving real-time, scalable, and nearly optimal planning methods remains an elusive ultimate goal. Real-world applications such as warehouse automation [1] demand methodologies capable of managing hundreds or more agents efficiently within a limited and short time frame. Consequently, the practical

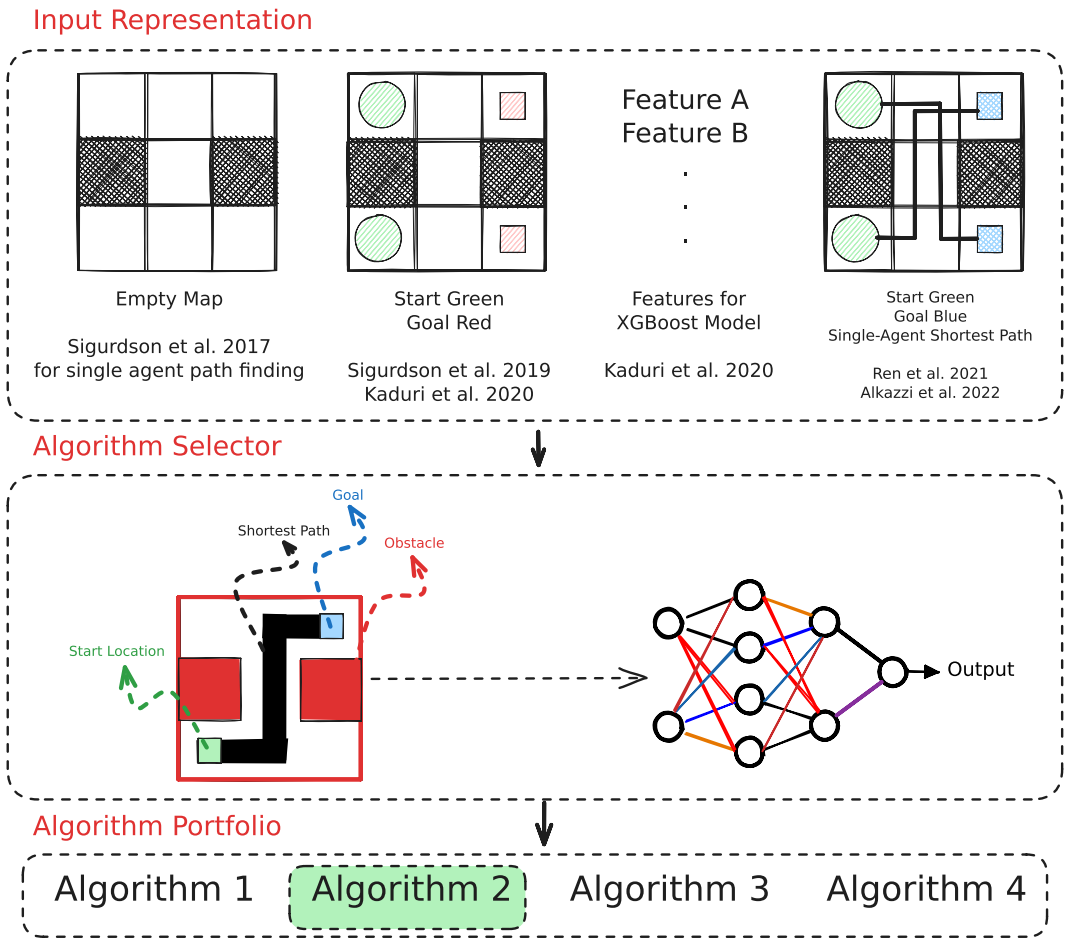


FIGURE 4. MAPF algorithm selection generic steps. Some example representations for MAPF instances are shown at the top. Using the given representation, a learning algorithm (i.e., selector; middle) selects the "best" MAPF algorithm from the available candidates (i.e., portfolio; bottom).

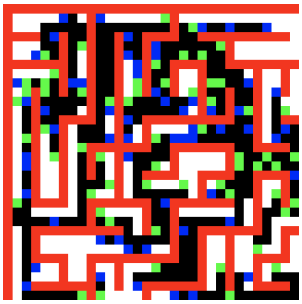


FIGURE 5. Overcrowded input representation from MAPFAST [65] dataset. The color scheme follows Figure 4.

deployment of MAPF algorithms still faces significant challenges.

Another challenge arises from the fact that, although centralized approaches, which dictate all agent actions based on complete information, have been extensively studied, real-world situations often require decentralized control systems, particularly in unstable network infrastructure environments. In these schemes, each agent independently decides its actions based on local observations. Furthermore, decentralized approaches are particularly valuable in dynamic environments with frequently changing obstacle layouts, requiring agents to adapt to these changes autonomously. While decentralized MAPF methods exist, as proposed by [118], [119], their reliance on complex, ad hoc rules often poses deployment challenges.

Two ML-assisted approaches have emerged in recent years to overcome these challenges:

- **Augmenting Existing Solvers**: Leverage ML to enhance the optimization process in traditional non-ML algorithms. The goal is to improve these solvers' efficiency, scalability, or adaptability.
- **Developing Effective Decentralized MAPF Methods**: Employ ML techniques like imitation or reinforcement learning to achieve decentralized agent behaviors. These behaviors are influenced by the agents' local observations and, if applicable, by their interactions with one another, to develop more responsive and self-governing decentralized MAPF approaches.

The subsequent parts will detail these strategies, representing pivotal progress in MAPF and opening new prospects for more efficient and adaptable path-finding solvers.

A. AUGMENTING EXISTING SOLVERS

The integration of ML into existing algorithms has been an active research topic. For instance, such integration has long been captivating in operations research, as shown in numerous studies [120], [121], [122], [123]. Recent developments in deep learning have further advanced these methods. Examples include learning to branch [124], [125], [126], [127], [128], [129], learning a node selection strategy [130], and learning to search from scratch [131].

Reflecting the progress in operations research, the incorporation of ML into MAPF algorithms presents a substantial opportunity for advancement in this field. The core idea primarily involves replacing traditional, manually crafted heuristics or components with counterparts derived through ML algorithms. These ML-based components are developed by training on large datasets generated through intensive offline computational processes. The following section elaborates on specific instances of this approach.

1) ENHANCING CONFLICT-BASED SEARCH

A widely recognized and exemplary optimal MAPF algorithm is Conflict-Based Search (CBS) [113]; hence, several studies enhance CBS with ML.

CBS derives a solution by employing a two-level search, typically referred to as a high- and low-level search. While the high-level search manages collisions between agents, the low-level search computes single-agent paths respecting constraints posed by the high-level. Once the low-level search returns paths, the high-level search checks for collisions in the proposed paths. When a collision (i.e., conflict) is detected, the high-level search divides the search space into two subspaces, one of which prohibits the use of the colliding location for one of the two agents in the collision, and the other prohibits the use of the location for the other agent. This division is done by constructing a binary tree structure called a conflict tree, where each node specifies agent spatiotemporal constraints. For example, if agents 1 and 2 collide at vertex E at timestep 1, one of the nodes created will prohibit agent 1 from being at vertex E at timestep 1. This process proceeds until finding collision-free paths.

There are various design choices in implementing CBS where ML can play a crucial role. For instance, selecting target conflicts to resolve significantly impacts the efficiency of the search [132], [133]. While conventional methods rely on manually designed strategies for conflict selection, Huang et al. [134] introduced a learning-based method for finding better target conflicts beyond handcrafted rules. Another example where ML helps CBS is in its suboptimal variant [135], which allows flexibility in selecting high-level nodes based on user-defined evaluation functions. For instance, [60] uses a support vector machine (SVM), and [63] employs a Transformer-based architecture [136] to learn effective heuristic functions that accelerate CBS, again beyond manually designed heuristics.

2) ENHANCING PRIORITIZED PLANNING

Prioritized planning [6], [137] is another widely-used MAPF algorithm focusing on planning based on assigned priorities per agent. It sequentially plans paths for each agent while avoiding conflicts with paths planned for previous higher-priority agents. The order in which agents are planned significantly impacts the success rate and the quality of the solution. Consequently, various studies have explored the effectiveness of different heuristics in determining these priority orders [138], [139], [140].

ML has been applied to enhance this prioritization process. For instance, Zhang et al. [61] adopted SVM to effectively learn and establish agent rankings. In contrast, Wang et al. [141] explored the use of genetic algorithms to synthesize priority functions.

3) ENHANCING OTHER MAPF SOLVERS

Viramni et al. [142] extends M* [18], a dynamic conflict resolution approach similar to CBS, with ML. Specifically, it incorporates imitation learning to develop individual agent policies aware of interactions among multiple agents. Huang et al. [62] extend the large neighborhood search for MAPF [143], which progressively improves the quality of solutions by modifying the paths of a subset of agents. In a nutshell, the work splits agents into subsets to be replanned, and a learned policy that decides which subset, if replanned, would lead to the most significant positive difference in the tracked metric. The recent work by Yan and Wu [144] uses a deep learning model to identify the effective subset to be selected.

4) CHALLENGES

The techniques mentioned above aim to accelerate MAPF algorithms. Generally, these approaches share a common challenge, which can be stated as follows:

Open Question 5: How can learning from experience be transferred from smaller to larger instances?
In other words, 'Is it possible to learn small-scale coordination patterns for large-scale systems?' [145]. Denser environments often exhibit behaviors not observed in settings with fewer agents, potentially leading to significant data distribution shifts. This shift can challenge ML models in adapting to such environments.

Furthermore, the enhancement of existing algorithms frequently raises the following question:

Open Question 6: How can we identify and extract effective features to maximize the performance of ML-assisted MAPF algorithms?
Hand-crafted features [60], [61], [62], [134] offer human-understandable insights and are preferred from an explainability perspective. However, this approach might

limit the exploration of other potentially valuable features. Several studies [63], [142] avoid the explicit use of hand-crafted features. However, finding the optimal feature extraction method often involves a process of trial and error.

B. LEARNING DECENTRALIZED POLICIES

In practical MAPF applications, the implementation of a centralized controller often faces significant challenges. These include unstable network environments, restricted network coverage, and dynamic environmental conditions that require robots to adapt based on sensory inputs. Although robust planning methodologies to tackle these issues are a focus of ongoing research, as will be elaborated in the upcoming section on execution, *decentralized* planning presents itself as a viable and promising solution. Under this approach, agents autonomously decide their actions, relying on local observations, improving adaptability and resilience in varying conditions.

Traditional decentralized approaches, akin to those outlined in works by [118] and [119], typically utilize complex, ad-hoc rules for action determination. This approach often makes them overly sensitive to specific problem formulations. In contrast, learning-based strategies offer a robust alternative. These involve the formulation of decentralized policies wherein each agent independently learns to execute actions that collaboratively steer the system toward the desired objective more efficiently. Modern edge devices' enhanced computational capabilities have made these learning-based methods increasingly feasible for real-world deployment.

This section thus explores diverse learning strategies that utilize ML to tackle the MAPF challenge from a decentralized perspective — specifically, the discussion centers on the advancements detailed in Table 1. We note that learning policies in *continuous* domains are out-of-scope of this review, i.e., studies where agents' actions are not restricted to graph structures such as [146], [147], and [148].

1) OVERVIEW

Generally, decentralized, learning-based approaches are structured to develop a policy that interprets surrounding information as input, typically represented as a *field-of-view (FOV)*. This policy then decides the action for the next timestep in a grid-world context, which could include: {UP, DOWN, LEFT, RIGHT, or WAIT}. Figure 6 illustrates this concept. These learning methodologies encompass the following paradigms:

- **Imitation learning (IL)** mimics ideal agents' behavior modeled by centralized MAPF algorithms. Optimal algorithms are often employed for demonstration purposes, but near-optimal and sub-optimal algorithms can also be utilized.
- **Reinforcement learning (RL)** develops a policy maximizing expected rewards. The reward structure typically includes objectives like collision avoidance and goal achievement with minimal actions. RL for MAPF is viewed as a decentralized partially observable Markov decision process [158].

The IL approach aids in stabilizing the learning process, whereas RL has the potential to discover more effective policies, potentially surpassing biases in existing MAPF algorithms. These approaches are not mutually exclusive; in fact, many studies integrate IL and RL in a hybrid fashion [67], [151], [154], [157]. In addition, some approaches employ **curriculum learning (CL)** [149], [153], [156], which feeds learning instances to IL/RL algorithms while gradually increasing the difficulty of the instances. This contributes to stabilizing the learning process.

The core perspective for these learning-based decentralized methods is aggregating information from the environment and other agents to achieve coordinated behaviors. Some methods are *observation-only*, in which each agents only see the locations of other agents within the FOV. This approach is attractive for deployments due to its minimal sensing assumptions. Meanwhile, other studies explicitly assume *communication* between agents to facilitate more sophisticated coordinated behaviors. When incorporating communication assumptions, three critical aspects must be considered: *when* to communicate, *whom* to communicate with, and *what* to communicate. Regarding the 'when,' all the studies we have reviewed employ communication at each timestep. Meanwhile, the 'whom' and 'what' vary significantly across different methods. Deterministic rules can be utilized for these purposes, but incorporating learning approaches, such as learning the contents of communication or identifying the optimal communication partners, also offers sensible options.

2) THE FIRST STUDY – PRIMAL

To provide a concrete example of learning-based decentralized approaches, we first illustrate a pioneering work named PRIMAL [67], which employs a decentralized policy for MAPF with the following technical components:

- **Observation:** The agent observation in PRIMAL consists of a square field-of-view (FOV) with four channels, representing binary maps of static obstacles (occupancy grid), positions of all agents, goals of neighboring agents, and the ego agent's own goal. These are depicted in Figure 7. Additionally, as input to its architecture, PRIMAL utilizes a vector directed from the ego agent to its goal, functioning essentially as a compass to help steer the agent towards its goal.
- **Aggregation of Other Agents' Information:** While PRIMAL is explained as a method without explicit communication, it implicitly assumes access to other agents' goal information, potentially through some oracle entity.
- **Action:** Agents execute discrete actions within the grid world: moving one cell in one of four directions or remaining in the current cell.
- **Learning Paradigm:** PRIMAL employs a dual approach combining imitation learning (IL) and

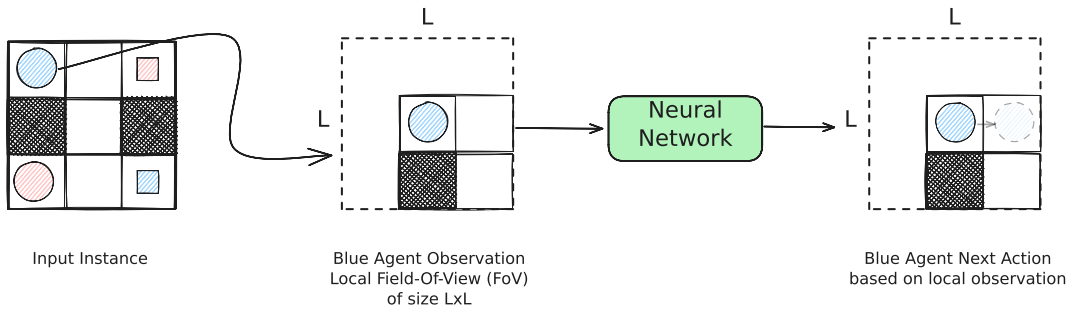


FIGURE 6. Generic view of learning-based decentralized approaches without communication.

TABLE 1. Overview of representative learning-based decentralized approaches. The existing studies rely on imitation learning (IL), reinforcement learning (RL), curriculum learning (CL), or a combination of these, as indicated in the learning column. These methods typically assume communication between agents, implicitly or explicitly, so that they can coordinate. The table summarizes 'what' to communicate with 'whom,' while in some methods, the agents rely on local observations only. Each method targets one-shot MAPF, lifelong variants, or both. In RL methods, reward design is critical to induce effective policies, and various schemes have been tested as summarized. The table uses '↓' for negative reward, '↑' for positive, '→' for the zero. Each method has typically been evaluated in terms of 'success rate' and 'path length' (i.e., flowtime), while some studies evaluate how many agents reach the goal without colliding. The table was partially inspired by [80].

			learning				communication		target		reward for RL										evaluation metrics					
	ref.	year	IL	RL	CL	methods	what	whom	one-shot	lifelong	move	wait (off goal)	wait (on goal)	collision	ego-agent goal	all-agent goal	subgoal	blocking	oscillation	off-route	success rate	path length	throughput	makespan	#goal reaching	collision rate
PRIMAL	[67]	19	✓	✓		A3C	(others' goals)	(all in FOV)	✓		↓	↓	→	↓	-	↑	-	↓	-	-	✓	-	-	-	-	-
GNN	[68]	20	✓			behavior cloning	encoded observation	all in FOV (k-hop)	✓		-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	✓	-
MAPPER	[149]	20		✓	✓	w/evolutionary alg.	**no communication**	-	✓		↓	↓	↓	↓	↑	-	-	-	↓	↓	✓	-	-	-	-	-
G2RL	[150]	20		✓		double DQN	**no communication**	-	✓		↓	-	-	↓	-	-	↑	-	-	-	✓	✓	-	-	✓	-
PRIMAL$_2$	[151]	21	✓	✓		A3C	(others' goals)	(all in FOV)	✓	✓	↓	↓	→	↓	↑	-	-	-	-	-	✓	✓	✓	-	-	-
MAGAT	[152]	21	✓			behavior cloning	encoded observation	all in FOV (k-hop)	✓		-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	✓	-
DHC	[153]	21		✓	✓	distributed IQL	encoded observation	max 2 agents in FOV	✓		↓	↓	→	↓	↑	-	-	-	-	-	✓	✓	-	-	-	-
PICO	[154]	22	✓	✓		A3C	encoded observation	comm. topology	✓		↓	↓	→	↓	-	↑	-	↓	-	-	✓	-	-	✓	-	-
DCC	[155]	22		✓	✓	distributed IQL	encoded obs. & relative pos.	selected agents	✓		↓	↓	→	↓	↑	-	-	-	-	-	✓	✓	-	-	-	-
SACHA	[156]	23		✓	✓	soft actor-critic	(cost-to-go) +encoded obs.	(all in FOV)	✓		↓	↓	→	↓	↑	-	-	-	-	-	✓	✓	-	-	-	-
SCRIMP	[157]	23	✓	✓		PPO	encoded observation	all in the map	✓		↓	↓	→	↓	→	-	-	↓	-	-	✓	-	-	✓	✓	✓
FOLLOWER	[69]	23		✓		PPO	**no communication**	-		✓	-	-	-	-	↑	-	↑	-	-	-	✓	-	✓	-	-	-

reinforcement learning (RL). During training, the model alternates between IL and RL for each step to refine the agent policy. IL is conducted through behavior cloning from an expert algorithm [18]. RL is performed with the Asynchronous Advantage Actor-Critic (A3C) method [159].

- **Reward:** In the RL phase, a substantial reward is given when *all* agents reach their goals, and a penalty is applied for each action taken other than staying in place. The reward structure also includes penalties for collisions and obstructing other agents' paths, thereby incentivizing agents to avoid hindering others' progress.

Following this seminal work, numerous learning-based MAPF methods have emerged. Most work uses the same action space as PRIMAL, but other technical components have variations. The subsequent sections discuss several distinctive approaches.

3) METHODOLOGY DETAILS

PRIMAL underwent further development in [151], leading to an evolved version named PRIMAL$_2$. This iteration not only builds upon the foundational principles of PRIMAL but also introduces enhancements such as an enriched observation space. Moreover, PRIMAL$_2$ has been specifically adapted for *lifelong* MAPF [2], an essential variant of MAPF in real-world settings. In lifelong MAPF, agents are immediately assigned new goals upon reaching their current ones, requiring uninterrupted, continuous navigation.

Communication-less approaches provide attractive alternatives for real-world deployment. Examples include MAPPER [149] and G2RL [150], which consider other agents as dynamic obstacles. Similar to PRIMAL, they incorporate an FOV in their observation space, but here, the locations of other agents are treated as dynamic obstacles. These methods predominantly rely on RL for training. In addition, MAPPER integrates an evolutionary algorithm to enhance

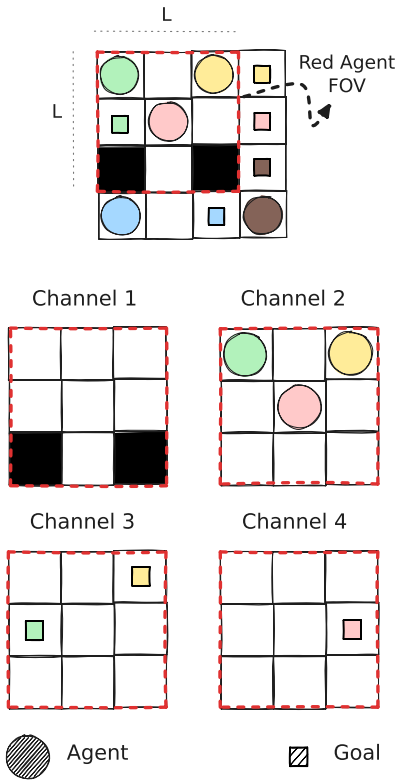


FIGURE 7. PRIMAL [67] observation representation considering a 3 × 3 agent FOV.

the refinement of agent policies. Another example of communication-free approaches can be seen in studies by Skrynnik et al. [69], [160], [161].

The application of graph neural networks (GNNs) in developing adaptive multi-robot systems represents an emerging and rapidly growing field of research. The application of GNNs to MAPF was first explored by Li et al. [68], [152] (i.e., GNN and MAGAT in Table 1). In their research, GNNs are employed to aggregate the intentions of other agents, supporting decision making. The inter-agent communications, learned through neural networks, involve encoded observations within the agents' FOV. It is noteworthy that these studies exclusively utilize IL as their learning strategy.

Not limited to GNNs, various techniques have been explored to integrate the intentions of other agents. SACHA [156] incorporates a cost-to-go map of other agents within the FOV, which extends beyond the single shortest path guidance employed in methods like MAPPER [149] or G2RL [150]. DHC [153] employs an attention-based architecture to learn communication strategies with agents within the FOV. Subsequent work [153] introduces selective communication, focusing on learning which agents to communicate with. PICO [154] also learns whom to communicate with, taking the form of communication topology, drawing inspiration from prioritized planning [6] for effective communication. In contrast, SCRIMP [157] assumes global communication among all agents, enabling successive planning while minimizing the FOV size to a 3 × 3 grid.

Policy learning in these systems predominantly employs RL. Techniques vary, with some using independent Q-learning (IQL), while others find success with soft actor-critic (SAC) or proximal policy optimization (PPO) [162].

a: REWARD SHAPING

The essence of RL is optimizing a policy through *rewards* as a response to agent actions. Table 1 presents the various reward schemes used in RL-based methods. These schemes typically include penalties for movement actions and collisions. While some methods provide rewards for achieving sub-goals, which are determined using heuristic search strategies, others introduce penalties for actions that hinder the progress of other agents (i.e., blocking), actions that lead to oscillatory behaviors (i.e., livelock), or actions that stray from the optimal path for a single agent (i.e., off-route). Notably, FOLLOWER [69] implements a straightforward reward structure, successfully avoiding complex reward shaping while achieving efficient agent coordination. This is a consequence of the effective utilization of planning with heuristic search.

4) EVALUATION METRICS

As we have seen, the attractiveness of learning-based MAPF methods is evident. However, there is a significant difference in the evaluation metrics used in various studies. Table 1 outlines how each study evaluates its respective method. The most commonly used metric is the *success rate* of the planning, which measures how frequently the solver can completely resolve the problem without any collisions. Quality of solutions is evaluated using metrics such as *path length* (i.e., flowtime), *makespan*, or *throughput* in lifelong MAPF, paralleling metrics used in traditional, non-learning-based MAPF methods. Additionally, some studies consider *the maximum number of goals agents can reach* within a given time limit, while others assess the *collision rate*, which quantifies the number of collisions with static obstacles or other agents. These criteria are particularly relevant to learning-based MAPF, as they are inherently inapplicable to centralized MAPF methods.

5) SIMULATOR ENVIRONMENTS

Learning paradigms heavily rely on simulated environments, and several factors should be given special attention, as described below:

- **Behavior upon reaching the goal:** Various approaches exist in handling agents' behavior upon reaching their goals. Some studies assume agents vanish upon arrival [151], others continually assign new targets [69], or the agent remains as a physical entity, capable of moving again to assist in unblocking other agents [67].
- **Agent's Possible Actions:** In certain studies, the ML models must choose from a list of actions, including invalid options such as actions that lead to collisions with

static obstacles [151], [153]. Conversely, other studies present the ML models only with a list of valid actions to choose from [67].

- **Field-of-View Construction:** Decentralized approaches typically emphasize partial observability, where each agent possesses a limited FOV surrounding its position. This assumption is driven by the aim to enable a more realistic and decentralized deployment in real-world scenarios. Agents' observations are usually categorized as either a bird's-eye-view (BEV) of the agent's surroundings [67], [151], [153], [155], [156], [157] or a first-person-view (FPV) based on sensor data [149], [163]. In a BEV setup, agents are obscured by obstacles like walls, but they would still be visible within the agent's FOV.
- **Synchrony of agent movement:** Most MAPF research assumes synchronous actions for each agent at every timestep. However, this assumption can be challenging for robot deployments. Therefore, the consideration of asynchronous movements may be beneficial.

These details significantly impact learning performance and benchmarking reliability, making the learning environment's uniformity crucial. While the MAPF benchmark [10] is commonly used for performance assessment, each study often relies on its unique environment to simulate agent dynamics, leading to varied underlying assumptions.

Open Question 7: Which benchmarking suite of environments and evaluation metrics would best reflect the performance of different techniques?
Considering all the differences in simulation environments and the diverse set of evaluation metrics, fairly comparing different techniques is challenging. Unifying those aspects would make benchmarking reliable and reproducible, helping faster progress in the field.

To support the development of learning-based MAPF methods, Figure 8 illustrates a brief list of open-source simulators that can serve as a reliable foundation. It is important to recognize that learning-based MAPF approaches necessitate scalable simulation environments, especially when simulating scenarios with hundreds or more agents. This requirement often excludes general-purpose environments like Unity, which facilitate 3D ML agents [164], due to their high resource demands.

- MiniGrid and MiniWorld [165] are efficient and user-friendly simulators initially developed for the BabyAI challenge. This challenge emphasizes grounded language learning through human interaction [166]. MiniWorld, in particular, allows for training models with reduced complexity while still offering the first-person view (FPV) observation for specific research. Both simulators can be slightly modified to accommodate MAPF tasks, enabling multi-agent simulation.
- Flatland [4] specifically addresses train pathfinding challenges, including situations involving train malfunctions. It achieves a performance of 156 FPS with 80 agents [167].
- RWARE [168] is designed for warehouse applications, tackling the Multi-Agent Pickup-and-Delivery problem [36]. It supports partial observability and has been reported to operate at approximately $1,255$ FPS in smaller settings with four agents.
- VMAS [169] distinguishes itself as the only vectorized framework for general-purpose Multi-Agent Reinforcement Learning (MARL) tasks featuring GPU acceleration.
- POGEMA [167] is purpose-built for addressing partially-observable MAPF challenges. It showcases an impressive capability of 83,000 FPS in scenarios with 80 agents. Moreover, POGEMA integrates with various MARL frameworks like PettingZoo [170], PyMARL [171], and SampleFactory [172]. This integration greatly lowers the barrier to entry for researchers and offers broad customization options.

6) CHALLENGES AND OPEN QUESTIONS

a: COMMUNICATION

The essence of solving MAPF involves integrating other agents' intentions into the ego-agent's decision-making process. Some methods employ *implicit* communication, depending solely on observing other agents' relative positions. Conversely, *explicit* communication strategies involve agents communicating at each timestep to coordinate their actions better.

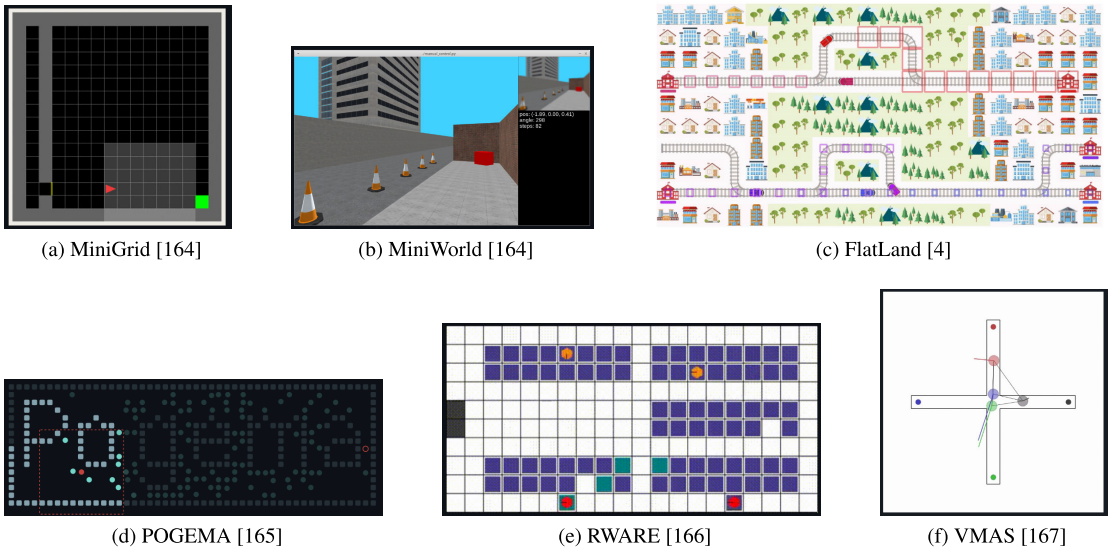
Open Question 8: Which communication strategy is most effective in real-world environments with their inherent challenges?
Real-world applications often encounter limitations due to the communication medium's speed, range, synchronicity, and vulnerability to external interference. These limitations may make certain learning-based methods impractical for real-world deployment. Therefore, it is crucial to consider these constraints when choosing a communication strategy for MAPF scenarios in real-life settings.

Implicit communication methods are appealing for these reasons, though mastering complex coordination remains a challenge. Moreover, communication might be restricted or prohibited in specific real-world situations for security reasons. This necessitates reliance solely on implicit communication for agent coordination.

Open Question 9: How can effectively learned implicit communication minimize the need and overhead of explicit communication while achieving comparable outcomes?
Studies such as [69] and [161] demonstrate that state-of-the-art decentralized behavior can be achieved by utilizing only local observations per agent without the need for explicit communication.

b: IMITATION LEARNING (IL)

While most approaches utilize IL, they are often confined to merely naive behavior cloning. In contrast, the ML literature presents more advanced methods, e.g., [173] and [174].



(a) MiniGrid [164] (b) MiniWorld [164] (c) FlatLand [4]

(d) POGEMA [165] (e) RWARE [166] (f) VMAS [167]

FIGURE 8. Snapshots of open-source simulation environments.

Open Question 10: How could more advanced IL methods improve the performance of agent-based approaches beyond naive behavior cloning (BC)?
BC is commonly employed due to its straightforward implementation. However, it shows a slight boost in performance compared to RL alone. Would more advanced techniques demonstrate a significant advantage over BC in MAPF scenarios?

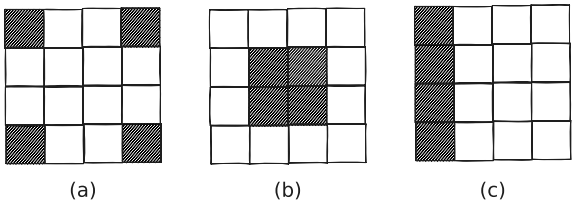
c: REINFORCEMENT LEARNING (RL)

As shown in Table 1, most RL-based approaches depend on complex reward shaping schemes currently necessary for stable learning. However, it is also appealing to learn effective policies using a simple reward structure, like AlphaZero [51] and its adaptation to decentralization MAPF [161], eliminating manual parameter tuning.

Open Question 11: How can we avoid reward shaping to eliminate human bias in the learning process?
Avoiding reward shaping allows the learning process to autonomously determine agents' behaviors, potentially leading to the discovery of innovative problem-solving strategies. Future methodologies could significantly benefit from incorporating such unbiased reward strategies.

d: CURRICULUM LEARNING (CL)

Considering that CL has often been employed to stabilize the learning process, it becomes clear that assessing the difficulty of MAPF instances and creating a dataset with progressively increasing difficulty levels would benefit the research community. Figure 9 displays three environments with identical obstacle densities, illustrating that Figure 9b, with its narrow corridors, may lead to unsolvable instances. In contrast, Figure 9a and 9c pose less significant challenges to MAPF algorithms. This indicates that simply generating



(a) (b) (c)

FIGURE 9. Environments with the same 25% obstacle density, but with different difficulty levels of the resulting MAPF instances.

random environments with varied obstacle densities does not guarantee the production of more challenging instances. Consequently, this raises the question of what factors truly determine the difficulty of a MAPF instance. The differences in environments, their impact on benchmarking, and the importance of diverse datasets are crucial challenges that hinder the development of more reliable and practical research in agent-based approaches.

Open Question 12: How can one construct a dataset of MAPF instances that progressively increases in difficulty?
By generating new datasets specifically designed for learning, we can develop highly adaptable models suitable for any given instance. This methodology was applied in the bipedal-walking-obstacle-course domain [175], where the generation of environmental challenges and the optimization of agents to overcome these challenges were successfully demonstrated. Such an approach facilitates the transfer of solutions across different problems, leading to more efficient problem-solving techniques. We argue that similar strategies could be effectively implemented for learning-based MAPF approaches. Ewing et al. [176] provide intriguing insights, suggesting that betweenness centrality can enhance optimal algorithms' performance prediction in specific MAPF instances. Another insight can be gained from [177], which suggests adjusting

the difficulty of MAPF instances by changing the maximum start-goal distance.

IV. EXECUTION

Once a MAPF solution is derived from the planning phase, applying this plan to real-world scenarios involves several challenges. These challenges partly stem from the need to establish a *robust* system capable of operating reliably amidst external disturbances or environmental uncertainties. Concurrently, there is a necessity for a *resilient* system that can restore its functionalities after encountering disruptions [49]. System disturbances manifest in various forms, including delays or inconsistencies in robot movements when executing the plan, kinodynamic constraints of robotic systems, random robot malfunctions, imperfect knowledge about the environment, road congestion caused by external agents like humans, and potential collisions or blockages due to unmodeled factors such as slips or friction, among others.

Researchers have tackled these challenges by creating two distinct approaches to execution strategies. The first type encompasses *offline* robust, proactive planning methods designed to handle uncertainty during execution. As a result, their solutions are almost deployment-ready, with minimal gaps between the planned outcome and real-world application. This direction includes MAPF formulations that allow agents to be delayed by a specific duration [178], or by any possible duration [45], agents-fault-ready planning [109], and multi-agent planning that assumes imperfect knowledge of the workspace [179], to name just a few. The second category is *online* replanning-based reactive methods for addressing unforeseen issues during execution. This encompasses methods for continuously synthesizing deadlock-free scheduling, assuming feedback-control mechanisms [42], [43], [44], [180], reactive planning that is tolerant to timing uncertainties [181], and minimum-effort replanning method from the original plan [182], among others.

While the aforementioned execution-related studies do not involve ML, the following discussion explores how ML can enhance them in several respects, including potential challenges. It is noteworthy that just a few ML approaches in MAPF specifically address the execution phase, highlighting a significant opportunity for development and improvement in this area.

A. TRAVEL AND ACTION TIME MODELING

ML can significantly benefit the execution phase by providing reasonable prediction models for robot motion time based on real-world environments. Such prediction models allow for the implicit incorporation of unknown complex real-world dynamics directly into the planning phase, facilitating a more dynamic and responsive strategy in handling potential conflicts and ensuring smoother, safer navigation of agents. For example, models predicting the expected delays in execution and movement for each robot type, based on historical data, can be incorporated during the planning phase. A notable study by Yu et al. [183] developed a deep learning model to predict potential delays each agent might face due to congestion along their designated routes, considering both historical congestion data and the pre-planned paths of agents for its forecasts. Another study by Ling et al. [184] employed reinforcement learning to plan drone paths in scenarios with uncertain travel times, addressing execution challenges during the planning phase. Kita et al. [185] proposed a CBS variant that updates the uncertainty model and adapts to the stochastic travel time, which is initially unknown, from collected data during execution. These studies demonstrate that traffic forecasting models can reduce the likelihood of unexpected deviations from planned paths, contributing to more reliable systems.

Not limited to these examples, ML techniques have proven to be useful in diverse forecasting tasks, such as reliably forecasting highway traffic [186], predicting short-term train delays [187], learning to estimate package pick-up arrival time [188], predicting the estimated time of arrival (ETA) in Google Maps [189], to name just a few. We anticipate that such ML techniques can be applied in the context of MAPF challenges to improve the solution execution efficiency and reliability. Meanwhile, this ML adaptation requires the following aspect.

Open Question 13: To what extent should the real-world agent dynamics captured by ML be reflected in MAPF?
Incorporating the full dynamics of real robots interacting with their environment, such as the friction of ground robots or the downwash effect of drones [9], enables us to derive solutions grounding to the environment. Such solutions minimize reality gaps, hence making them execution-ready. Meanwhile, this makes MAPF significantly more challenging because these environmental factors escalate MAPF to more multi-robot motion planning, which is a tremendously difficult problem even if we limit the scope of finding feasible solutions [190], [191]. Consequently, attempting to represent the comprehensive dynamics seems implausible for obtaining a scalable solution in MAPF. Instead, a more pragmatic approach is to adopt certain levels of abstraction. For example, building uncertainty models of the travel time between two locations may be more effective. This balances the need for realism in representing agent dynamics with the practical constraints of MAPF algorithms.

B. FAILURE PREDICTION

Forecasting potential failures by ML, as seen in failure prediction for railway rolling stock equipment [192], allows for preemptive adjustments to the paths of agents, thereby reducing the likelihood of collisions and disruptions. For example, we can consider combining fault prediction models with agent fault-tolerant MAPF plans [109] to prevent cascade crashes in advance, where a set of paths for each agent is precomputed for each potential failure. Another potential application of failure prediction models is the design of a fault-tolerant environment that minimizes the possibility

of agent failures in MAPF systems, which is achieved in combination with the environment optimization methods discussed in Section II-B.

Open Question 14: How can ML enhance the fault tolerance of MAPF systems?

Exploring the enhancement of fault tolerance in Multi-Agent Path Finding (MAPF) systems through machine learning is vital, particularly for the future of automation infrastructure. Machine learning's role in predicting failures is a starting point, but broader attention is needed to bolster MAPF systems' resilience and dependability.

V. CONCLUSION

This paper presented a comprehensive overview of how machine learning (ML) techniques can enhance Multi-Agent Path Finding (MAPF), a pivotal technology in the modern and forthcoming era of automation. Our review categorizes MAPF-related technologies into three dimensions: *(i)* representation, which involves modeling the workspace; *(ii)* planning, the process of determining agents' actions; and *(iii)* execution, the implementation of the planned outcomes. In each area, we discussed how ML can help MAPF become more practical, scalable, and robust for deployment beyond the current limitations of non-learning-based methods. We revisit the structure of our work, mentioning the leaner assortment of studies in the representation and execution sections, which reflects the current scope of research in these areas. This detail, while subtle, underscores our commitment to presenting a well-rounded and thoroughly curated overview of the field, aligning with the evolving dynamics of current research trends.

By detailing current ML-based approaches for solving MAPF problems, this review enables researchers to rapidly assess their potential strengths and weaknesses. It also highlights new research avenues through the open questions we propose. Although this is an emerging field, we envision and look forward to MAPF, supported by ML, shaping the future infrastructure of large-scale automated systems.

REFERENCES

[1] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectr.*, vol. 45, no. 7, pp. 26–34, Jul. 2008.

[2] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, pp. 11272–11281.

[3] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[4] S. Mohanty, E. Nygren, F. Laurent, M. Schneider, C. Scheller, N. Bhattacharya, J. Watson, A. Egli, C. Eichenberger, C. Baumberger, G. Vienken, I. Sturm, G. Sartoretti, and G. Spigler, "Flatland-RL: Multi-agent reinforcement learning on trains," 2020, *arXiv:2012.05893*.

[5] J. Li, Z. Chen, Y. Zheng, S.-H. Chan, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Scalable rail planning and replanning: Winning the 2020 flatland challenge," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 31, May 2021, pp. 477–485.

[6] D. Silver, "Cooperative pathfinding," in *Proc. AAAI Conf. Artif. Intell. Interact. Digital Entertainment (AIIDE)*, 2005, pp. 117–122.

[7] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, and S. Koenig, "Planning, scheduling and monitoring for airport surface operations," in *Proc. AAAI Workshop, Planning Hybrid Syst.*, 2016, pp. 1–7.

[8] A. Okoso, K. Otaki, and T. Nishi, "Multi-agent path finding with priority for cooperative automated valet parking," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2135–2140.

[9] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 856–869, Aug. 2018.

[10] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Bartak, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. Annu. Symp. Combinat. Search (SoCS)*, 2019, pp. 151–158.

[11] D. Ratner and M. Warmuth, "Finding a shortest solution for the NxN extension of the 15-puzzle is intractable," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 1986, pp. 168–172.

[12] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2010, pp. 1261–1263.

[13] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2013, pp. 1443–1449.

[14] J. Yu, "Intractability of optimal multirobot path planning on planar graphs," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 33–40, Jan. 2016.

[15] J. Banfi, N. Basilico, and F. Amigoni, "Intractability of time-optimal multirobot path planning on 2D grid graphs with holes," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 1941–1947, Oct. 2017.

[16] T. Geft and D. Halperin, "Refined hardness of distance-optimal multi-agent path finding," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2022, pp. 1–8.

[17] H. Ma, C. Tovey, G. Sharon, T. Kumar, and S. Koenig, "Multi-agent path finding with payload transfers and the package-exchange robot-routing problem," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 1–8.

[18] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, Feb. 2015.

[19] P. Surynek, "Unifying search-based and compilation-based approaches to multi-agent path finding through satisfiability modulo theories," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 202–203.

[20] S. D. Han and J. Yu, "Integer programming as a general solution methodology for path-based optimization in robotics: Principles, best practices, and applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1890–1897.

[21] J. Li, D. Harabor, P. J. Stuckey, H. Ma, G. Gange, and S. Koenig, "Pairwise symmetry reasoning for multi-agent path finding search," *Artif. Intell.*, vol. 301, Dec. 2021, Art. no. 103574.

[22] E. Lam, P. Le Bodic, D. Harabor, and P. J. Stuckey, "Branch-and-cut-and-price for multi-agent path finding," *Comput. Oper. Res.*, vol. 144, Aug. 2022, Art. no. 105809.

[23] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 7643–7650.

[24] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artif. Intell.*, vol. 310, Sep. 2022, Art. no. 103752.

[25] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, pp. 12353–12362.

[26] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, pp. 10256–10265.

[27] K. Okumura, "LaCAM: Search-based algorithm for quick multi-agent pathfinding," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2023, pp. 11655–11662.

[28] K. Okumura, "Improving LaCAM for scalable eventually optimal multi-agent pathfinding," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 1–14.

[29] Z. Chen, D. Harabor, J. Li, and P. J. Stuckey, "Traffic flow optimisation for lifelong multi-agent path finding," 2023, *arXiv:2308.11234*.

[30] T. T. Walker, N. R. Sturtevant, and A. Felner, "Extended increasing cost tree search for non-unit cost domains," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 534–540.

[31] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artif. Intell.*, vol. 305, Apr. 2022, Art. no. 103662.
[32] K. Kasaura, M. Nishimura, and R. Yonetani, "Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2D roadmaps," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10494–10501, Oct. 2022.
[33] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4608–4615, Jul. 2021.
[34] K. Okumura and X. Défago, "Quick multi-robot motion planning by combining sampling and search," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 1–11.
[35] H. Zhang, S.-H. Chan, J. Zhong, J. Li, P. Kolapo, S. Koenig, Z. Agioutantis, S. Schafrik, and S. Nikolaidis, "Multi-robot geometric task-and-motion planning for collaborative manipulation tasks," *Auto. Robots*, vol. 47, no. 8, pp. 1537–1558, Dec. 2023.
[36] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2017, pp. 1–9.
[37] J. Svancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták, "Online multi-agent pathfinding," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 7732–7739.
[38] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2016, pp. 1–8.
[39] W. Honig, S. Kiesel, A. Tinka, J. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2018, pp. 1–9.
[40] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 1–8.
[41] K. Okumura and X. Défago, "Solving simultaneous target assignment and path planning efficiently with time-independent execution," *Artif. Intell.*, vol. 321, Aug. 2023, Art. no. 103946.
[42] M. Cap, J. Gregoire, and E. Frazzoli, "Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5113–5118.
[43] W. Hönig, T. K. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," in *Proc. Int. Conf. Automated Planning Scheduling (ICAPS)*, 2016, pp. 477–485.
[44] H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 1–8.
[45] K. Okumura, F. Bonnet, Y. Tamura, and X. Défago, "Offline time-independent multiagent path planning," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2720–2737, Aug. 2023.
[46] B. Shen, Z. Chen, M. Aamir Cheema, D. D. Harabor, and P. J. Stuckey, "Tracking progress in multi-agent path finding," 2023, *arXiv:2305.08446*.
[47] K. Okumura, "Engineering LaCAM*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding," 2023, *arXiv:2308.04292*.
[48] Financial Times. (2021). *Ocado's Largest Warehouse Shut After Robot Collision Causes Fire*. [Online]. Available: https://www.ft.com/content/aaddf4b1-a78b-4289-b42f-fd3f5cd7f176
[49] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," 2021, *arXiv:2109.12343*.
[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[51] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.
[52] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, Jan. 2018.
[53] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, and F. Fuchs, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022.
[54] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zidek, and A. Potapenko, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10674–10685.
[56] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Tech. Rep., 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
[57] J. D. M.-W. C. Kenton and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2019, pp. 1–16.
[58] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, Jan. 2022, Art. no. eabk2822.
[59] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.
[60] T. Huang, B. Dilkina, and S. Koenig, "Learning node-selection strategies in bounded-suboptimal conflict-based search for multi-agent path finding," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2021, pp. 1–8.
[61] S. Zhang, J. Li, T. Huang, S. Koenig, and B. Dilkina, "Learning a priority ordering for prioritized planning in multi-agent path finding," in *Proc. Int. Symp. Combinat. Search*, Jul. 2022, pp. 208–216.
[62] T. Huang, J. Li, S. Koenig, and B. Dilkina, "Anytime multi-agent path finding via machine learning-guided large neighborhood search," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, pp. 9368–9376.
[63] C. Yu, Q. Li, S. Gao, and A. Prorok, "Accelerating multi-agent planning using graph transformers with bounded suboptimality," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 3432–3439.
[64] D. Sigurdson, V. Bulitko, S. Koenig, C. Hernandez, and W. Yeoh, "Automatic algorithm selection in multi-agent pathfinding," 2019, *arXiv:1906.03992*.
[65] J. Ren, V. Sathiyanarayanan, E. Ewing, B. Senbaslar, and N. Ayanian, "MAPFAST: A deep algorithm selector for multi agent path finding using shortest path embeddings," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2021, pp. 1–9.
[66] J.-M. Alkazzi, A. Rizk, M. Salomon, and A. Makhoul, "MAPFASTER: A faster and simpler take on multi-agent path finding algorithm selection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 10088–10093.
[67] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.
[68] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 11785–11792.
[69] A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov, "Learn to follow: Lifelong multi-agent pathfinding with decentralized replanning," in *Proc. PRL Workshop Ser. Bridging Gap Between AI Planning Reinforcement Learn.*, 2023, pp. 1–17.
[70] F. F. Arias, B. Ichter, A. Faust, and N. M. Amato, "Avoidance critical probabilistic roadmaps for motion planning in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 10264–10270.
[71] K. Okumura, R. Yonetani, M. Nishimura, and A. Kanezaki, "CTRMs: Learning to construct cooperative timed roadmaps for multi-agent path planning in continuous spaces," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2022, pp. 1–12.
[72] A. Felner, R. Stern, S. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proc. Annu. Symp. Combinat. Search (SoCS)*, 2017, pp. 29–37.
[73] R. Stern, "Multi-agent path finding—An overview," in *Artificial Intelligence* (Lecture Notes in Computer Science), vol. 11866, Oct. 2019, pp. 96–115. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-33274-7_6
[74] O. Salzman and R. Stern, "Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2020, pp. 1711–1715.

[75] H. Ma, "Graph-based multi-robot path finding and planning," *Current Robot. Rep.*, vol. 3, no. 3, pp. 77–84, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s43154-022-00083-8
[76] S. Lin, A. Liu, J. Wang, and X. Kong, "A review of path-planning approaches for multiple mobile robots," *Machines*, vol. 10, no. 9, p. 773, Sep. 2022.
[77] K. S. Yakovlev, A. Andreychuk, A. Skrynnik, and A. I. Panov, "Planning and learning in multi-agent path finding," *Doklady Math.*, vol. 106, pp. 79–84, Dec. 2022.
[78] R. Zhou, "Research of the methods on multi-agent path finding," *Highlights Sci., Eng. Technol.*, vol. 39, pp. 1131–1140, Apr. 2023.
[79] J. Chung, J. Fayyad, Y. Al Younes, and H. Najjaran, "Learning team-based navigation: A review of deep reinforcement learning techniques for multi-agent pathfinding," 2023, *arXiv:2308.05893*.
[80] J. Gao, Y. Li, X. Li, K. Yan, K. Lin, and X. Wu, "A review of graph-based multi-agent pathfinding solvers: From classical to beyond classical," *Knowl.-Based Syst.*, vol. 283, Jan. 2024, Art. no. 111121.
[81] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, Jul. 1998.
[82] J. P. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2008, pp. 1928–1935.
[83] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. 14th Int. Symp. (ISRR)*, 2011, pp. 3–19.
[84] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, Mar. 1986.
[85] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Trans. Robot. Autom.*, vol. 8, no. 1, pp. 23–32, Feb. 1992.
[86] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, p. 4282, 1995.
[87] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[88] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017.
[89] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 3420–3431.
[90] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.
[91] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.
[92] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
[93] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 18–47, Jan. 2014.
[94] S. Choudhury, O. Salzman, S. Choudhury, and S. S. Srinivasa, "Densification strategies for anytime motion planning over large dense roadmaps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3770–3777.
[95] B. Saund and D. Berenson, "Fast planning over roadmaps via selective densification," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2873–2880, Apr. 2020.
[96] C. Henkel and M. Toussaint, "Optimized directed roadmap graph for multi-agent path finding using stochastic gradient descent," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 776–783.
[97] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, Sep. 1951.
[98] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[99] X. Gao, Z. Pan, and R. Ni, "Multi-robot path planning in complex environments via graph embedding," 2021, *arXiv:2108.03368*.
[100] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–9.
[101] C. Doersch, "Tutorial on variational autoencoders," 2016, *arXiv:1606.05908*.
[102] Z. Gao and A. Prorok, "Environment optimization for multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 3440–3446.
[103] Z. Gao and A. Prorok, "Constrained environment optimization for prioritized multi-agent navigation," *IEEE Open J. Control Syst.*, vol. 2, pp. 337–355, 2023.
[104] D. Vainshtein and O. Salzman, "Multi-agent terraforming: Efficient multi-agent path finding via environment manipulation," in *Proc. Annu. Symp. Combinat. Search (SoCS)*, 2021, pp. 239–241.
[105] D. Vainshtein, Y. Sherma, K. Solovey, and O. Salzman, "Terraforming—Environment manipulation during disruptions for multi-agent pickup and delivery," 2023, *arXiv:2305.11510*.
[106] Y. Zhang, M. C. Fontaine, V. Bhatt, S. Nikolaidis, and J. Li, "Multi-robot coordination and layout design for automated warehousing," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2023, pp. 1–8.
[107] Y. Zhang, M. C. Fontaine, V. Bhatt, S. Nikolaidis, and J. Li, "Arbitrarily scalable environment generators via neural cellular automata," in *Proc. Conf. Neural Inf. Process. Syst.*, 2023, pp. 1–14.
[108] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. (2020). *Growing Neural Cellular Automata*. Distill. [Online]. Available: https://distill.pub/2020/growing-ca
[109] K. Okumura and S. Tixeuil, "Fault-tolerant offline multi-agent path planning," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2023, pp. 11647–11654.
[110] H. Zender, G.-J. M. Kruijff, and I. Kruijff-Korbayova, "A portfolio approach to algorithm select," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2003, pp. 1542–1543.
[111] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Frechette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren, "Aslib: A benchmark library for algorithm selection," *Artif. Intell.*, vol. 237, pp. 41–58, Aug. 2016.
[112] O. Kaduri, E. Boyarski, and R. Stern, "Algorithm selection for optimal multi-agent pathfinding," in *Proc. Int. Conf. Automated Planning Scheduling (ICAPS)*, 2020, pp. 161–165.
[113] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, Feb. 2015.
[114] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T. K. S. Kumar, and S. Koenig, "Adding heuristics to conflict-based search for multi-agent path finding," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 28, Jun. 2018, pp. 83–87.
[115] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient sat approach to multi-agent path finding under the sum of costs objective," in *Proc. Eur. Conf. Artif. Intell. (ECAI)*, 2016, pp. 810–818.
[116] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[117] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.
[118] A. Wiktor, D. Scobee, S. Messenger, and C. Clark, "Decentralized and complete multi-robot motion planning in confined spaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 1168–1175.
[119] Y. Zhang, K. Kim, and G. Fainekos, "DisCoF: Cooperative pathfinding in distributed systems with limited sensing and communication range," in *Proc. Int. Symp. Distrib. Robotic Syst. (DARS)*, 2016, pp. 325–340.
[120] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 1995, pp. 1114–1120.
[121] J. A. Boyan, "A reinforcement learning framework for combinatorial optimization," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 1996, p. 1379. [Online]. Available: https://cdn.aaai.org/AAAI/1996/AAAI96-230.pdf
[122] T. Stockheim, M. Schwind, and W. Koenig, "A reinforcement learning approach for supply chain management," in *Proc. Eur. Workshop Multi-Agent Syst.*, 2003, pp. 1–13. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fda8416d1e3cab07a645ff9f2c79e6fba43c88d8#:~:text=The%20rein%2D%20forcement%20learning%20algorithm,machine%20following%20the%20queue's%20schedule
[123] S. Chang, "A survey on reinforcement learning in global optimization," UC Berkeley, Tech. Rep., 1998. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=70f579c4482607d5fa51d111240ca75d1f644332
[124] E. B. Khalil, P. L. Bodic, L. Song, G. L. Nemhauser, and B. N. Dilkina, "Learning to branch in mixed integer programming," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 1–8.
[125] D. Sorokin and A. Kostin, "TreeDQN: Learning to minimize branch-and-bound tree," 2023, *arXiv:2306.05905*.
[126] C. W. F. Parsonson, A. Laterre, and T. D. Barrett, "Reinforcement learning for branch-and-bound optimisation using retrospective trajectories," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2022, pp. 4061–4069.

[127] L. Scavuzzo, F. Chen, D. Ch'etelat, M. Gasse, A. Lodi, N. Yorke-Smith, and K. I. Aardal, "Learning to branch with tree MDPs," in *Proc. Conf. Neural Inf. Process. Syst.*, 2022, pp. 18514–18526.

[128] Q. Qu, X. Li, and Y. Zhou, "Yordle: An efficient imitation learning for branch and bound," 2022, *arXiv:2202.01896*.

[129] Q. Qu, X. Li, Y. Zhou, J. Zeng, M. Yuan, J. Wang, J. Lv, K. Liu, and K. Mao, "An improved reinforcement learning algorithm for learning to branch," 2022, *arXiv:2201.06213*.

[130] A. G. Labassi, D. Ch'etelat, and A. Lodi, "Learning to compare nodes in branch and bound with graph neural networks," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 32000–32010.

[131] J. Song, R. Lanka, A. Zhao, A. Bhatnagar, Y. Yue, and M. Ono, "Learning to search via retrospective imitation," 2018, *arXiv:1804.00846*.

[132] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and E. Shimony, "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2015, pp. 223–225.

[133] E. Boyarski, A. Felner, P. Le Bodic, D. D. Harabor, P. J. Stuckey, and S. Koenig, "F-aware conflict prioritization & improved heuristics for conflict-based search," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, pp. 12241–12248.

[134] T. Huang, S. Koenig, and B. Dilkina, "Learning to resolve conflicts for multi-agent path finding with conflict-based search," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2021, pp. 11246–11253.

[135] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. Annu. Symp. Combinat. Search (SoCS)*, 2014, pp. 19–27.

[136] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–11.

[137] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 477–521, Nov. 1987.

[138] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2005, pp. 430–435.

[139] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robot. Auto. Syst.*, vol. 41, pp. 89–99, Nov. 2002.

[140] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-robot path deconfliction through prioritization by path prospects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 9809–9815.

[141] S. Wang, V. Bulitko, T. Huang, S. Koenig, and R. Stern, "Synthesizing priority planning formulae for multi-agent pathfinding," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)*, 2023, pp. 360–369.

[142] L. Virmani, Z. Ren, S. Rathinam, and H. Choset, "Subdimensional expansion using attention-based learning for multi-agent path finding," 2021, *arXiv:2109.14695*.

[143] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "Anytime multi-agent path finding via large neighborhood search," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2021, pp. 4127–4135.

[144] Z. Yan and C. Wu, "Neural neighborhood search for multi-agent path finding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024, pp. 1–22.

[145] A. Prorok, J. Blumenkamp, Q. Li, R. Kortvelesy, Z. Liu, and E. Stump, "The holy grail of multi-robot planning: Learning to generate online-scalable solutions from offline-optimal experts," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2021, pp. 1–8.

[146] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 285–292.

[147] B. Rivière, W. Hönig, Y. Yue, and S.-J. Chung, "GLAS: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4249–4256, Jul. 2020.

[148] S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme, "Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning," in *Proc. Annu. Conf. Robot Learn. (CoRL)*, 2022, pp. 576–586.

[149] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 11748–11754.

[150] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6932–6939, Oct. 2020.

[151] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning—Lifelong," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 6, no. 2, pp. 2666–2673, Apr. 2021.

[152] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5533–5540, Jul. 2021.

[153] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8699–8705.

[154] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang, "Multi-agent path finding with prioritized communication learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 10695–10701.

[155] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1455–1462, Apr. 2022.

[156] Q. Lin and H. Ma, "SACHA: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 5100–5107, Aug. 2023.

[157] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "SCRIMP: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 9301–9308.

[158] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 1994, pp. 157–163.

[159] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1928–1937.

[160] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "When to switch: Planning and learning for partially observable multi-agent pathfinding," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Aug. 31, 2023, doi: 10.1109/TNNLS.2023.3303502.

[161] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. Panov, "Decentralized Monte Carlo tree search for partially observable multi-agent pathfinding," 2023, *arXiv:2312.15908*.

[162] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[163] Z. Liu, Q. Liu, L. Tang, K. Jin, H. Wang, M. Liu, and H. Wang, "Visuomotor reinforcement learning for multirobot cooperative navigation," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3234–3245, Oct. 2022.

[164] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.

[165] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. Samuel Castro, and J. Terry, "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks," 2023, *arXiv:2306.13831*.

[166] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, "BabyAI: First steps towards grounded language learning with a human in the loop," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=rJeXCo0cYX

[167] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "POGEMA: Partially observable grid environment for multiple agents," 2022, *arXiv:2206.10944*.

[168] C. Filippos, L. Schäfer, and S. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 10707–10717.

[169] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "VMAS: A vectorized multi-agent simulator for collective robot learning," in *Proc. Int. Symp. Distrib. Robotic Syst. (DARS)*, 2022, pp. 42–56.

[170] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, and R. Perez-Vicente, "PettingZoo: Gym for multi-agent reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 15032–15043.

[171] M. Samvelyan, T. Rashid, C. Schroeder de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," 2019, *arXiv:1902.04043*.

[172] A. Petrenko, Z. Huang, T. Kumar, G. S. Sukhatme, and V. Koltun, "Sample factory: Egocentric 3D control from pixels at 100000 FPS with asynchronous reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 7652–7662.

[173] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1–9.

[174] G. Swamy, S. Choudhury, J. A. Bagnell, and Z. S. Wu, "Of moments and matching: A game-theoretic framework for closing the imitation gap," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 10022–10032.

[175] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions," 2019, *arXiv:1901.01753*.

[176] E. Ewing, J. Ren, D. Kansara, V. Sathiyanarayanan, and N. Ayanian, "Betweenness centrality in multi-agent path finding," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2022, pp. 1–9.

[177] T. Phan, J. Driscoll, J. Romberg, and S. Koenig, "Confidence-based curriculum learning for multi-agent path finding," 2024, *arXiv:2401.05860*.

[178] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N.-F. Zhou, "Robust multi-agent path finding and executing," *J. Artif. Intell. Res.*, vol. 67, pp. 549–579, Mar. 2020.

[179] B. Shofer, G. Shani, and R. Stern, "Multi agent path finding under obstacle uncertainty," in *Proc. Int. Conf. Automated Planning Scheduling (ICAPS)*, 2023, pp. 402–410.

[180] A. Berndt, N. Van Duijkeren, L. Palmieri, and T. Keviczky, "A feedback scheme to reorder a multi-agent execution schedule by persistently optimizing a switchable action dependency graph," 2020, *arXiv:2010.05254*.

[181] K. Okumura, Y. Tamura, and X. Défago, "Time-independent planning for multiple moving agents," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2021, pp. 11299–11307.

[182] J. Kottinger, T. Geft, S. Almagor, O. Salzman, and M. Lahijanian, "Introducing delays in multi-agent path finding," 2023, *arXiv:2307.11252*.

[183] G. Yu and M. T. Wolf, "Congestion prediction for large fleets of mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7642–7649.

[184] J. Ling, T. Gupta, and A. Kumar, "Reinforcement learning for zone based multiagent pathfinding under uncertainty," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 30, Jun. 2020, pp. 551–559.

[185] A. Kita, N. Suenari, M. Okada, and T. Taniguchi, "Online re-planning and adaptive parameter update for multi-agent path finding with stochastic travel times," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2023, pp. 1–9.

[186] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial–temporal synchronous graph convolutional networks: A new framework for spatial–temporal network data forecasting," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2020, pp. 914–921.

[187] B. Gao, L. Zhang, D. Ou, and D. Dong, "A novel deep learning model for short-term train delay prediction," *Inf. Sci.*, vol. 645, Oct. 2023, Art. no. 119270.

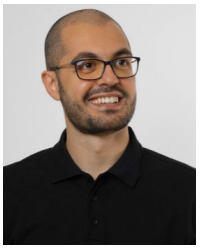
[188] H. Wen, Y. Lin, F. Wu, H. Wan, Z. Sun, T. Cai, H. Liu, S. Guo, J. Zheng, C. Song, and L. Wu, "Enough waiting for the couriers: Learning to estimate package pick-up arrival time from couriers' spatial–temporal behaviors," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 3, pp. 1–22, Jun. 2023.

[189] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velickovic, "ETA prediction with graph neural networks in Google maps," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 3767–3776.

[190] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the 'warehouseman's problem,'" *Int. J. Robot. Res.*, vol. 3, no. 4, pp. 76–88, Dec. 1984.

[191] P. Spirakis and C. K. Yap, "Strong np-hardness of moving many discs," *Inf. Process. Lett.*, vol. 19, no. 1, pp. 55–59, Jul. 1984.

[192] L. De Simone, E. Caputo, M. Cinque, A. Galli, V. Moscato, S. Russo, G. Cesaro, V. Criscuolo, and G. Giannini, "LSTM-based failure prediction for railway rolling stock equipment," *Exp. Syst. Appl.*, vol. 222, Jul. 2023, Art. no. 119767.



JEAN-MARC ALKAZZI received the B.Eng. and M.Eng. degrees in software engineering from Antonine University, in 2016 and 2018, respectively, and the Ph.D. degree in computer science from Université Franche-Comté, France, in 2023.

From 2019 to 2021, he led projects with the BMW Innovation Lab, Munich, where he implemented machine learning and operations research solutions to enhance production line efficiency. Since 2021, he has been leading the Applied AI Team, IDEALworks GmbH, a BMW Group spin-off. His work revolves around multi-robot systems, fleet management systems, and applied research in industrial settings. His research interest includes coordinating heterogeneous multi-robot systems within logistics facilities. He is dedicated to pushing the boundaries of AI in industrial applications, aiming to develop a state-of-the-art fleet management system capable of managing complex logistics operations at scale. His work resulted in BMW winning the "Pioneer in AI in Production" Award, in 2020. For more details visit https://blog.jeremarc.com/.

KEISUKE OKUMURA (Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees in computer science from Tokyo Institute of Technology (TokyoTech), Tokyo, Japan, in 2018, 2020, and 2023, respectively.

In 2022, he was a Visiting Scholar with Sorbonne University, France. Since 2023, he has been a Research Scientist with the National Institute of Advanced Industrial Science and Technology (AIST), Japan. He has been a Visiting Scholar with the University of Cambridge. His research interests include designing intelligent collective behavior for swarm agents, in particular, the development of multi-agent planning methods tailored to large-scale automation. He has been honored with several research awards, including the Best Student Paper Award from ICAPS 2022 and the Ph.D. Dissertation Award from Tokyo Institute of Technology, in 2024. For more details visit https://kei18.github.io/.