Attention-based Priority Learning for Limited Time Multi-Agent Path Finding

Yibin Yang*
Tsinghua University
Beijing, China
yyb19@mails.tsinghua.edu.cn

Mingfeng Fan*
Central South University
Changsha, China
mingfan2001@gmail.com

Chengyang He
National University of Singapore
Singapore, Republic of Singapore
chengyanghe@u.nus.edu

Jianqiang Wang
Tsinghua University
Beijing, China
wjqlws@tsinghua.edu.cn

Heye Huang†
Tsinghua University
Beijing, China
hhy18@mails.tsinghua.edu.cn

Guillaume Sartoretti
National University of Singapore
Singapore, Republic of Singapore
guillaume.sartoretti@nus.edu.sg

ABSTRACT

Solving large-scale Multi-Agent Path Finding (MAPF) within a limited time remains an open challenge, despite its importance for many robotic applications. Recent learning-based methods scale better than conventional ones, but remain suboptimal and often exhibit low success rates within a limited time on large-scale instances. These limitations often stem from their black-box nature. In this study, we propose a hybrid approach that incorporates prioritized planning with learning-based methods to explicitly address these challenges. We formulate prioritized planning as a Markov Decision Process and introduce a reinforcement learning-based prioritized planning paradigm. In doing so, we develop a novel Synthetic Score-based Attention Network (S2AN) to learn conflict/blocking relationships among agents, and deliver blocking-free priorities. By integrating priority mechanisms and leveraging a new attention-based neural network for enhanced multi-agent cooperative strategies, our method enhances solution completeness while trading off scalability and maintains linear time complexity, thus offering a robust avenue for large-scale MAPF tasks. Comparisons demonstrate its superiority over current learning-based methods in terms of solution quality, completeness, and reachability within limited time constraints, especially in large-scale scenarios. Moreover, an extensive set of numerical results reveals superior completeness compared to restricted-time Priority-Based Search (PBS) and Priority Inheritance with Backtracking (PIBT) in medium to large-scale obstacle-dense scenarios.

KEYWORDS

Multi-Agent Path Finding; Attention-Based Network; Reinforcement Learning; Prioritized Planning

ACM Reference Format:
Yibin Yang, Mingfeng Fan, Chengyang He, Jianqiang Wang, Heye Huang, and Guillaume Sartoretti. 2024. Attention-based Priority Learning for Limited Time Multi-Agent Path Finding. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

*Both authors contributed equally to the paper
†Corresponding author

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Multi-Agent Path-Finding (MAPF) [1, 23] is a central multi-agent/-robot problem, in which multiple agents have to plan collision-free paths to their pre-assigned goals while minimizing total travel times. There has been an increasing interest in large-scale MAPF due to its broad applications in autonomous vehicles [12], multi-robot systems [6, 19], and video games [16]. However, due to the curse of dimensionality, solving large-scale MAPF in a limited time still remains a significant challenge [9, 28]. For instance, in gaming scenarios, real-time computation and rendering of paths for a large number of agents are necessary to enable them to find collision-free paths quickly and maintain player engagement. In this work, we focus on large-scale MAPF, and present an approach that aims to strike an efficient balance between high completeness and computing times, by boosting the performance of prioritized planning through machine learning.

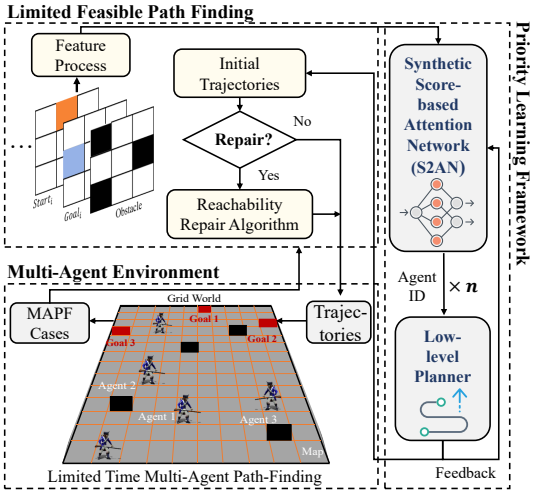


Figure 1: Limited Time MAPF Problem and Attention-based Priority Learning Framework.

MAPF algorithms can be categorized into non-learning and learning approaches. Non-learning algorithms can be further divided into systematic search algorithms, rule-based algorithms, and prioritized algorithms. Systematic search algorithms, such as CBS [22], offer optimal or bounded-suboptimal solutions. However, due to the NP-Hard nature of MAPF problems, they face challenges when handling large-scale scenarios and exhibit exponential-time in the worst case. On the other hand, rule-based algorithms guarantee finding a solution in polynomial time but may encounter planning time or memory issues in practical applications. Prioritized algorithms, such as MAPF-LNS2 [11] (the state-of-the-art unbounded suboptimal MAPF algorithms) and PBS [14], perform well in practice despite being incomplete. They decompose the MAPF problem into a series of single-agent planning problems that can be solved sequentially based on some underlying priority, each treating higher-priority agents as obstacles. While low-level planning can achieve optimality, it does not guarantee the optimality of the overall plan, as high-level planners often lack the ability to adequately consider global features and predict interactions with lower-priority agents. In contrast, learning-based methods have shown great abilities at leveraging global information and achieving high performances in many other applications, such as routing problems [8]. In particular, existing learning-based MAPF approaches [17, 21, 27] often rely on distributed reinforcement learning (RL) or partially centralized Multi-Agent Reinforcement Learning (MARL) algorithms. They scale well across different team sizes, world sizes, and obstacle densities. However, these methods cannot guarantee optimality or completeness. Due to their black-box nature, learning-based methods lack behavior consistency and low-level optimality. This often leads to extended time requirements for guiding all agents to their goals. In practice, learning methods struggle to achieve optimality and may fail to solve problems within a limited time, especially in highly obstructed, small-world-size, or dense-agent scenarios [17, 21].

To address these limitations, this paper introduces a novel paradigm for Multi-Agent Path Finding (MAPF) that combines learning and prioritized planning. We harness the power of machine learning to reason about global information and directly generate high-quality priorities for the agents. These priorities are then used to quickly find near-optimal low-level paths for the agents, and thus solve the problem with high completeness and low time complexity. In doing so, our approach directly learns cooperation in the priority space of the agents. Specifically, we employ an encoder to encode the MAPF instance and a decoder to sequentially output the agent to be planned next, based on the encoded information and information about all planned trajectories so far. To the best of our knowledge, we are the first to utilize Reinforcement Learning (RL) for learning priorities for MAPF. Accordingly, the main contributions are outlined as follows.

(1) We introduce a novel hybrid paradigm for MAPF, which combines machine learning and priority-based planning. There, the behavior inconsistency and suboptimal low-level planning challenges are explicitly addressed by building upon the prioritized planning framework, also offering promising scalability. Experiments demonstrate that our method has better solution quality and higher completeness than existing learning-based methods in highly obstructed scenarios.
(2) Our framework is able to yield high-completeness solutions to large-scale MAPF instances under strong time constraints, typically within seconds for teams of up to 180 agents in 32*32 random 20% obstacle maps. It significantly outperforms a variety of MAPF algorithms in terms of success rate on maps with large teams and high-density obstacles, while keeping time complexity linear.

2 RELATED WORK

2.1 MAPF Algorithms

Numerous algorithms have been developed to solve MAPF problem, which is focused on finding the minimum-cost, conflict-free solution [6, 15]. Optimal MAPF resolution is NP-complete, frequently encountering scalability issues [4]. Optimal MAPF resolution is NP-complete, frequently encountering scalability issues [4]. Bounded suboptimal solvers ensure that the cost of the solution is within a given bound of the optimal value, and they operate at a relatively accelerated pace [2]. However, they cannot handle large-scale problems. Unbounded suboptimal solutions offer rapid computational speed and demonstrate admirable performance in large-scale agent scenarios, finding applications in both research and industry [2]. Accordingly, this paper concentrates on the elucidation of unbounded suboptimal solutions. Unbounded suboptimal methods for solving MAPF can be categorized into two broad classifications: learning-based and non-learning-based approaches.

Classic non-learning-based approaches. These approaches generally include rule-based algorithms [18, 20, 26], bounded-suboptimal algorithms with an infinite bound factor [2, 24] and prioritized planning algorithms. Rule-based algorithms generally offer completeness guarantees and can be executed in polynomial time under certain (weakened) conditions. However, in practical applications, these methods often struggle with global optimality and are time-consuming for large-scale problems. Bounded-suboptimal algorithms are based on CBS. While these methods no longer require boundedness, their foundation in CBS techniques still presents challenges when addressing larger-scale problems. Prioritized planning algorithms have gained significant attention due to their simplicity and efficiency. They can be seamlessly extended with other algorithms and exhibit strong performance.

Learning-based approaches. These methods represent another paradigm for computing approximate solutions to the MAPF problem through the iterative refinement of temporal strategies informed by feedback rewards. These approaches engage in policy planning for each agent, employing local observations or messages to determine the next action. Notable methods in this category include distributed RL techniques such as PRIMAL [21], MARL approaches like SCRIMP [27] and PICO [13]. PRIMAL initially introduces RL and learns a fully decentralized policy in a partially observable environment. Recent research has emphasized communication learning for the MAPF problem. While specific features and networks enhance success rates, they often struggle to scale to large scenarios due to communication complexity. Learning-based methods show scalability under certain conditions. However, due to the challenges associated with multimodal policy learning and

inconsistent behavior, they struggle to find feasible solutions for scenarios involving hundreds or more agents.

2.2 Prioritized Planning

Prioritized planning algorithms are neither complete nor optimal; however, they represent a computationally inexpensive and highly effective approach to solving the MAPF problem [3, 7, 14]. These methods are predicated on a straightforward prioritization scheme: each agent is assigned a unique priority, and agents are subsequently ordered according to predefined global priorities. The agent with the highest priority is enabled to compute its individual optimal path first. The Attention routing problem shares similarities with MAPF and is also NP-Hard [8, 10], requiring the generation of a non-repeating sequence. It necessitates a trade-off between solution quality and computational time [5]. Mainstream approaches can be classified into three types: improvement, backtracking, and construction methods.

Backtracking. These methods often necessitate iterative traversal of the entire priority space, exemplified by algorithms like PBS [14]. Such methods iteratively search through a subset of the space until a feasible solution is discovered. Analogous to CBS [22], this method partitions the space based on conflicts arising from the generated paths and is provably p-complete.

Iteration. The iteration methods proceed to generate progressively superior solutions, building upon previous ones, until completion. MAPF-LNS2 [11] employs the large neighborhood search framework for its iterative procedure and currently represents the state-of-the-art in unbounded MAPF methods.

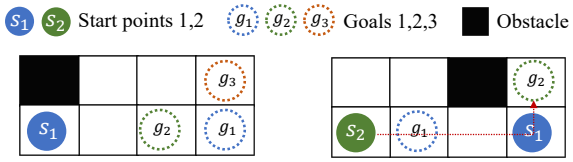
Construction. The construction methods typically produce a solution in a single pass, contrasting with improvement and backtracking approaches, which require repetitive iteration and search traversal. Hence, construction methods boast superior real-time capabilities alongside a commendable success rate. It is well known in the routing problems [8, 10] but is rare in MAPF. They are capable of rapidly generating high-quality suboptimal solutions and exhibit admirable scalability. Accordingly, this paper introduces a construction-type attention routing method.

3 PRELIMINARIES

3.1 Multi Agent Path Finding

Definition 1 (Multi-Agent Path Finding). In a known 2D static grid map containing multiple obstacles, we consider a scenario with n agents. Each agent possesses a distinct starting location that does not overlap with obstacles or other agents' starting positions, as well as unique goal locations that are similarly obstacle-free and non-overlapping. At each discrete time step, agents can move to neighboring cells or remain stationary at their current locations. The primary objective is for each agent to plan a path that satisfies the following conditions: 1) Begins at their respective starting positions, 2) Avoids collisions with other agents and obstacles, and 3) all agents end up at their goal locations at some terminal time step. The solution to this problem comprises a set of paths for all agents. The optimization objective is to minimize the sum of the path lengths. However, instead of seeking the optimal solution, our focus is on efficiently finding a feasible solution within a limited time frame.

Many decoupled Multi-Agent Path Finding (MAPF) algorithms break down the multi-agent planning problem into multiple smaller constrained single-agent planning problems. These single-agent planners have been extensively studied and are typically governed by non-learning methods. One such powerful single-agent planner is SIPPS (Safe Interval Path Planning with Soft constraints) [11], capable of efficiently handling scenarios involving moving obstacles. We employ the SIPPS as the low-level single-agent path planner to complete our framework.



(a) Failure: Target blocking (b) Failure: Run over

Figure 2: Priority Failure Reasons.

3.2 Prioritized Planning Failure Analysis

Prioritized planning is a prevalent decoupled method for MAPF. It operates by assigning a fixed global order and then planning the agents' paths sequentially. The highest-priority agent begins by optimally planning its path while avoiding static obstacles. Subsequently, each agent plans its path, considering not only static obstacles but also treating previously planned agents' paths as moving obstacles. This process repeats n times, where n is the number of agents, ultimately solving the problem. This method efficiently simplifies the multi-agent problem into a series of single-agent problems.

Prioritized planning, while widely used, has been proven to be incomplete and suboptimal. Randomly assigned priorities can lead to poor or even failed solutions. Failures in prioritized planning can be categorized into two main situations [11], as illustrated in Figure 2. The first situation is goal blocking, where higher-priority agents (e.g., a_2 and a_3) reach their goals early and occupy them, entirely preventing lower-priority agents (e.g., a_1) from reaching their destinations. While feasible solutions can be generated when a_1 has higher priority than a_2 or a_3, in dense obstacle environments, agents can be blocked by different agents at various priority levels, necessitating multiple visits to their goal before completing the task. This makes it challenging to find a feasible priority assignment in a single pass.

The second reason is run over, where higher-priority agents (e.g., a_2) plan paths that intersect with lower-priority agents (e.g., a_1). In such cases, a_1 lacks the required maneuvering space to plan a collision-free path. It is worth noting that if all agents can wait at starts and goals without obstructing others, any priority assignment would solve the problem. We refer to this as a well-formed problem.

4 METHOD

This section outlines our approach to the prioritized problem within a Markov Decision Process (MDP) framework. We start by formulating the problem as an MDP, subsequently highlighting two critical and time-efficient features that enable our learning-based

model to learn patterns and relationships within the data pertaining to MAPF problems. Next, we elaborate on our policy network, S2AN, which extracts latent information from these critical features and learns to deliver priority sequences with high completeness in an encoding-decoding manner. Lastly, we present our reinforcement learning technique, which efficiently trains this model.

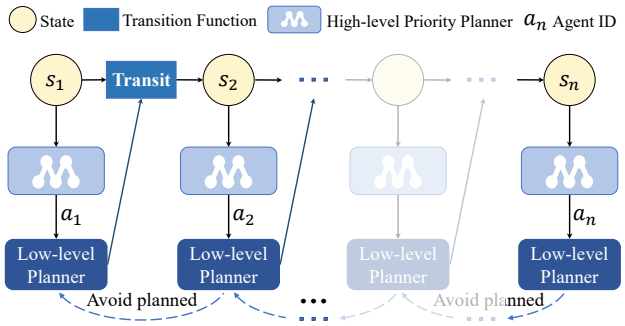


Figure 3: MDP Formulation of Prioritized Planning.

4.1 Markov Decision Process Formulation

A MAPF instance comprises three basic components: an obstacle map, start locations, and goal locations. We describe the MDP of prioritized planning in Figure 3. In the initial state s_1, the high-level priority planner selects an agent, such as a_1, as its action. Subsequently, the low-level single-agent path planner computes the path for this agent while considering the paths of previously planned agents as soft-moving obstacles. The low-level planner provides feedback to the environment, leading to a transition to state s_2 based on this feedback. This iterative process repeats for n times, ultimately progressing the state to its final configuration. Given an instance with n agents, we formally define the MDP of prioritized planning as follows.

State. The state consists of two parts, a static and a dynamic one. The static part contains the basic components of the instance, i.e., the obstacle map, start, and goal locations, as well as additional static features such as harmful features. On the other hand, the dynamic part contains the agents that have been selected before and their respective planned paths.

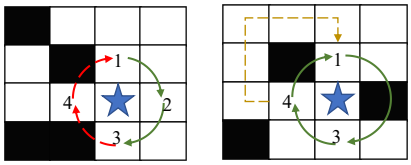
Action. The action of the MDP is the index of an agent, i.e., selecting the agent at the current time step. Since the high-level planner cannot choose an agent more than once, the legal set of actions consists of all the agents that have not been chosen.

State transition. We use SIPPS as the low-level path solver. Once the policy network delivers an action, SIPPS generates the corresponding agent's path. This planned path constitutes as a component of the dynamic state of the next state.

Reward. The goal is to find a feasible solution. We achieve this by minimizing the collisions. When new collisions are detected, $R(s_t, a_t)$ are equal to the negative number of new collision agents. When the episode terminates and a collision-free solution is found, we give a positive episode reward.

4.2 Feature Design

Effective feature engineering is the key to unlocking the full potential of a learning-based model for solving MAPF problems. In our paper, we employ two well-designed features including *harmful feature* and *target matrix*, based on domain knowledge to capture the essential information from data given a MAPF instance. *Harmful feature* assesses local connectivity to determine whether an agent's goal is blocking its neighborhood or not. Figure 4 offers an illustrative example of how this feature distinguishes whether an agent's goal is harmful or not. In practice, goal blocking is a common occurrence. When an agent's goal obstructs other agents, its priority should be lowered. However, manually detecting all blocking pairs can be time-consuming. Consequently, we aim to autonomously discern these blocking relationships by utilizing the provided *harmful feature* in a learning-based model. The *harmful feature* allows the model to capture the blocking relationships inherent in the data, enhancing its ability to address the MAPF problem effectively.



(a) harmful goal (b) not harmful goal

Figure 4: Harmful Feature Illustration. The star indicates the goal location. Assume the goal has up to four non-obstacle neighbors 1,2,3,4. We examine the connectivity sequentially. If any of them fail, then the goal is blocking a region and is considered harmful. In sub-figure (a), 3 cannot find a path to 4 so it is harmful. In (b), all the non-obstacle neighbors are connected so it is not harmful.

On the other hand, another feature *target matrix* is derived from the Minimal Target Visit A* algorithm (MTVA*), inspired by MAPF-LNS2 [11]. In the original A* algorithm, the search prioritizes exploring the node with the minimal f value from the open set to find a path. f is the sum of the node's heuristic distance to the goal and its traveled distance. However, in MTVA*, instead of focusing solely on the minimal f value, it explores the node with the least visits to other agents' goals, denoted as tv. In cases where multiple nodes $|N|$ share the minimum tv values, it will select the node in $|N|$ with the smallest f value. We aggregate visiting pairs and represent them as *target matrix*. Formally, $TargetMatrix_{ij} = 1$ if agent i passes through agent j's goal during the execution of MTVA*; otherwise, $TargetMatrix_{ij} = 0$. Notably, *target matrix* is a dynamic feature that changes whenever the high-level planner generates an action. Hence, we integrate *target matrix* into the decoding step to provide auxiliary knowledge to our learning-based model. Similar to the algorithm for topological sorting with zero in-degrees, when an agent no longer obstructs others, it can be selected as the highest-priority agent among the remaining ones.

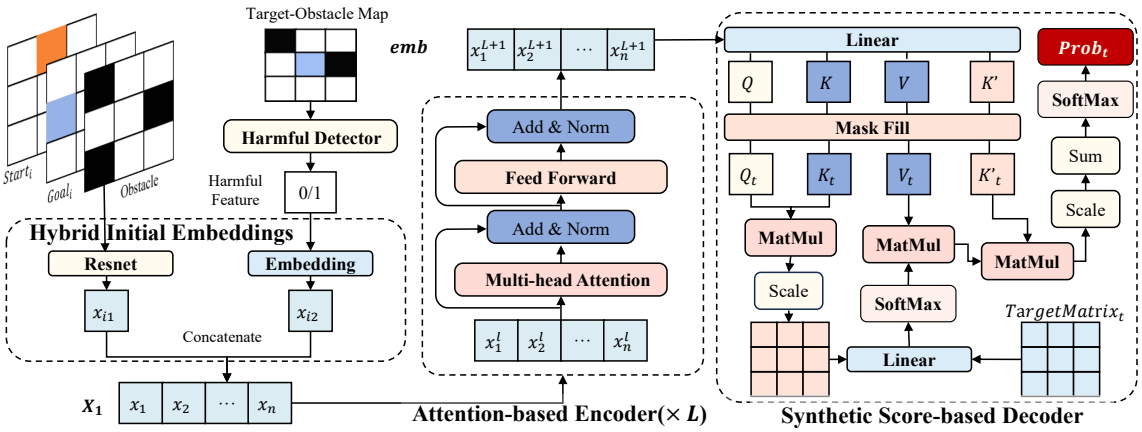


Figure 5: The Network Structure of Synthetic Score-based Attention Network(S2AN).

4.3 Policy Network

To solve the MAPF problem, we develop a novel learning-based method namely S2AN. Our S2AN is built upon a Transformer-style [25] architecture and consists of hybrid initial embeddings, attention-based encoder, and synthetic score-based decoder, as shown in Figure 5.

Hybrid Initial Embeddings. The hybrid initial embeddings serve as inputs to S2AN and are obtained by mapping the raw features associated with a MAPF instance into a high-dimensional feature space. For a given instance $\mathcal{S}$ with n agents, each agent possesses two types of raw features: instance features (comprising start location, goal location, and obstacle map) and a domain-specific feature *harmful feature*. We process these two types of features separately. Specifically, we employ a ResNet to project instance features into a d_1-dim vector denoted as x_{i1}. On the other hand, we utilize a linear layer to map *Harmful feature* which is a boolean variable into a d_2-dim vector represented as x_{i2}. Subsequently, these two vectors, x_{i1} and x_{i2}, are concatenated to form the d-dim hybrid initial embedding x_i of agent i.

Attention-based Encoder. The attention-based encoder takes the hybrid initial embeddings $X_1 = \{x_i\}_1^n$ as inputs and generates an advanced embedding matrix denoted as *emb* for all agents. We employ an encoder structure similar to that of Transformer [25], comprising L attention layers. Each attention layer consists of a multi-head attention (MHA) sublayer that facilitates communication between agents to capture their relationships more effectively and a feed-forward (FF) sublayer that consists of two linear projection layers with a ReLU activate function in between. Each sublayer is followed by a skip-connection and batch normalization(BN). Let X_l represent the input to attention layer l ($l \in \{1, \cdots, L\}$). The core of the MHA sublayer, based on X_l, is formally defined as follows:

$$Q_l^h, K_l^h, V_l^h = W_{Q,l}^h X_l, W_{K,l}^h X_l, W_{V,l}^h X_l, \tag{1}$$

$$\begin{aligned} A_l^h &= Atten(Q_l^h, K_l^h, V_l^h) \\ &= Softmax(Q_l^h * (K_l^h)^T / \sqrt{d_k}) V_l^h, h = 1, 2, \cdots, H, \end{aligned} \tag{2}$$

$$MHA(Q_l, K_l, V_l) = Concat(A_l^1, A_l^2, ..., A_l^H) W_O, \tag{3}$$

$$\hat{h_l} = BN(X_l + MHA(Q_l, K_l, V_l)), \tag{4}$$

where Q_l^h, K_l^h, and V_l^h represent *Query*, *Key*, and *Value* matrices, respectively; H is the number of attention heads; $W_{Q,l}^h, W_{K,l}^h, W_{V,l}^h \in R^{d_k \times d}$ with $d_k = d/H$, and $W_O \in R^{d \times d}$ are trainable parameters. Eqs.(1) and (2) are applied for each attention head h. Then the output of the MHA sublayer, i.e., h_l, is fed into the FF sublayer to get the output of the attention layer l, i.e., X_{l+1}, expressed as Eq.(5). After applying these operations for a total of L attention layers, we ultimately obtain the attention-based encoder's output, $emb = X_{L+1}$, which serves as the input to the synthetic score-based decoder.

$$X_{l+1} = BN(\hat{h_l} + FF(\hat{h_l})), \tag{5}$$

Synthetic Score-based Decoder. Based on the embedding matrix *emb*, our synthetic score-based decoder constructs solutions to MAPF problems in an autoregressive manner. The decoder consists of one synthetic attention layer that integrates a domain-specific feature,i.e., *target matrix*, into an MHA mechanism, and one compatibility layer. We first calculate the *Query*, *Key*, and *Value* matrixes, denoted as Q^h, K^h, and V^h, respectively, for the synthetic attention layer, and single head key termed K' for the compatibility layer through linear transformation, expressed as follows,

$$Z = W_Z emb, Z \in \left\{Q^h, K^h, V^h, K'\right\}. \tag{6}$$

Given the constraint that each agent can be selected no more than once, we should forbid the actions representing selected agents to be chosen again during the decoding process. To achieve this, we explicitly eliminate the visited agents in the instance at each step. Specifically, at each time step t, we update Q_t^h, K_t^h, V_t^h, and K_t', where Q_1^h, K_1^h, V_1^h, and K_1' are initialized as Q^h, K^h, V^h, and K', respectively. This update is achieved by applying a mask fill layer that sets the corresponding information of the selected agents to zero in the original Q^h, K^h, V^h, and K', as shown below,

$$Z_t = MaskFill(Z, mask_t), Z \in \left\{Q^h, K^h, V^h, K'\right\}, \tag{7}$$

where $mask_t$ is a boolean tensor, and its value corresponding to the selected agent is set to zero. In the synthetic attention layer, it first computes multi-head self-attention score α_{self}^h, i.e.,

$\alpha^h_{self} = Q^h_t (K^h_t)^T / \sqrt{d_k}$, $h = 1, 2, \cdots, H$. Thereafter, we introduce the domain-specific feature *target matrix* as our auxiliary attention score which further facilitates various feature embeddings. Notably, the *target matrix*, i.e., $TargetMatrix_t$ ($TargetMatrix_1 = TargetMatrix$), is a dynamic feature about the blocking pairs, which also be updated by the mask fill layer. Then, the self-attention score α^h_{self} and *target matrix* are concatenated and fed into a linear layer to obtain the synthetic attention score, i.e., $\alpha^h = W_{synth}(Concat(\alpha^h_{self}, TargetMatrix_t))$, where $W_{synth} \in R^{2H \times H}$ is a trainable parameter. The synthetic attention score is then normalized to $\tilde{\alpha}^h$ through SoftMax, which is further used to calculate attention values for each head, i.e., $head^h = \tilde{\alpha}^h V^h_t$. Subsequently, all heads are concatenated together, which then passes through a linear layer with a trainable parameter $W_d \in R^{d \times d}$ to obtain the output of the synthetic attention layer, i.e., $Q_p = Concat(head^1, \cdots, head^H) W_d$. Finally, we obtain the probability of selecting actions via the compatibility layer, i.e., $Prob_t = Softmax(\Sigma^n_{i=1}(Q_p K^{'T}_t / \sqrt{d_k}))$.

4.4 Training Algorithm

The S2AN model is trained using the REINFORCE algorithm [29], which incorporates an entropy term to encourage exploration during training. The REINFORCE loss is defined as the negative of expectation of reward, i.e., $\mathcal{L}_{RL} = -\mathbb{E}_{\tau \sim \pi_\theta}[R]$. Then, the gradient for minimizing the REINFORCE loss $\mathcal{L}_{RL}$ is defined as follows,

$$\nabla_\theta \mathcal{L}_{RL} = -\mathbb{E}_\tau [\sum_{t=1}^{T} G_t \cdot \nabla_\theta log \pi_\theta(a_t|s_t)], \tag{8}$$

where the T is the total timesteps and equal to the number of agents n; τ is the selected action sequence (i.e. the solution) generated by the policy π_θ; G_t is the return at time t, which is the discounted sum of rewards. Additionally, an entropy loss is introduced to prevent the model from converging to a suboptimal policy too quickly, defined as follows,

$$\mathcal{L}_{entropy} = -Entropy(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\pi_\theta(a|s_t) log(\pi_\theta(a|s_t))]. \tag{9}$$

Finally, the total gradient $\nabla_\theta \mathcal{L}$, is a combination of the REINFORCE loss and the entropy loss, i.e., $\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}_{RL} + k_{entropy} \nabla_\theta \mathcal{L}_{entropy}$, where the $k_{entropy}$ is the coefficient of entropy loss to adjust the importance of the entropy regularization during training, striking a balance between exploration and exploitation.

In this paper, we employ two types of action-selection strategies: a *greedy* strategy that always selects the action with the maximum probability and a *sampling* strategy that samples actions based on their probabilities. During the training process, we utilize the *sampling* strategy to encourage exploration. In contrast, during the inference process, we employ the *greedy* strategy to efficiently find a feasible solution to the MAPF problem.

4.5 Reachability Repair Algorithm

In traditional non-distributed methods, the planned trajectory is often either entirely feasible or plagued by collisions and cannot be executed, presenting a binary outcome. On the other hand, distributed methods, especially learning-based methods, frequently yield collision-free solutions. When they fail to find a feasible solution, they will output a collision-free solution, while not every agent arrives at its goal. This characteristic is essential in certain applications. However, our method is rooted in the prioritized planning architecture. To address the limitation, we find it necessary to provide a fast and effective post-processing method. In scenarios where a collision solution is encountered, we aim to eliminate collisions and enable the maximum number of agents possible to reach their intended goals.

To address this issue, we propose a reachability repair algorithm for S2AN, as shown in the appendix. The core idea of this method involves replanning paths for all colliding agents, ensuring they avoid collisions with others instead of reaching their goals. This will be achieved by iteratively running SIPPS for the colliding agents, and adjusting goals to start points to avoid collisions. An obvious advantage is that the time complexity does not increase significantly since most non-colliding agents can maintain their initial paths. Specifically, we first decode the actions in a greedy manner, resulting in obtaining the initial solution P, where $P = (a_1, P_{a1}) \cup (a_2, P_{a2}) \cup ... \cup (a_n, P_{an})$. Then, we select all the colliding agents set C in the solution P. If there are no collisions, a feasible solution will be found and returned as P. Otherwise, the solution needs to be repaired. Furthermore, since we change the goals of colliding agents, some unmodified agents might share the same goal as the modified ones, making the MAPF case invalid. Therefore, it is critical to repeatedly adjust agents that collide with the modified agent's goal until no further collisions occur. Subsequently, we remove the corresponding agents' paths in P and then replan their paths using SIPPS to find a minimal number of collisions from their start points to the new goals. If any agent fails to find a collision-free path, the repair method will return an empty solution. If repair is successful, the reach rate can be calculated as $P_{reach} = |C_L|/n$, where $|C_L|$ is the number of agents that altered their goals. Thus, this method still has a linear time complexity and improves the reach rate of S2AN.

5 EXPERIMENTS

In our research, we conducted an extensive experimental evaluation for three aspects. 1) We compared the success rates of various MAPF algorithms, under identical maximum planning times. This is done to ascertain the high completeness of the proposed method in relation to other prevalent algorithms. 2) We aimed to compare the success rate, episode length, and reach rate against current learning methods to present the advantages. 3) We embarked on an ablation study to critically evaluate the effectiveness of a salient component of our algorithm. For the purpose of these tests, we generated 100 randomized maps and scenarios. These were constructed with an obstacle density of 20 percent, derived from the standard MAPF benchmark suite. To further underscore the superiority of our priority learning framework, an additional 100 random scenes were generated using empty maps.

5.1 Settings and Metrics

Settings. In the training procedure, the batch size is set to 64, and Ray is employed to accelerate our sampling process. For the initial embedding network, we utilize ResNet18 to encode the maps. In both the encoder and decoder networks, we incorporate 8 attention

heads with 128-dimensional features. The encoder consists of 2 layers. In the encoder and decoder network, we use 8 attention heads with 128 dimension features. The encoder layer number is 2. In terms of the learning procedure, we employ the Adam optimizer with a learning rate of 3×10^{-4}. The entropy weight is set at 0.3, and the discount factor γ, is set to 0.95. The training is executed on a server equipped with a 4x3090 GPU and an i9-10980XE CPU, but only one GPU is used during the training process. Source codes are in https://github.com/marmotlab/S2AN.

Metrics. In our evaluation of performance, we focus on five key metrics: success rate, reach rate, makespan, episode length, and runtime. These metrics serve to benchmark our approach against other methods within the generated scenarios.

5.2 Limited Time Completeness Experiment

We select two representative algorithms for comparison to validate our high completeness within a limited time. 1) PIBT [18].This algorithm, based on priority inheritance, boasts linear time complexity, which is on par with our S2AN algorithm. It operates at high speed and provides a guarantee of reachability. It stands as a convincing benchmark due to its matching time complexity. 2) Same time constraint PBS [14]. PBS is a well-known prioritized planning method due to its simplicity and efficiency. This priority-based search algorithm with backtracking is capable of exploring all possible priorities given sufficient time. To provide a fair comparison, we evaluate it using the same time constraint applied in our algorithm. To ensure a more robust evaluation of the algorithm's effectiveness, we calculate the success rates of the PBS algorithm when subjected to time limits of 0.5T and 1.5T respectively.

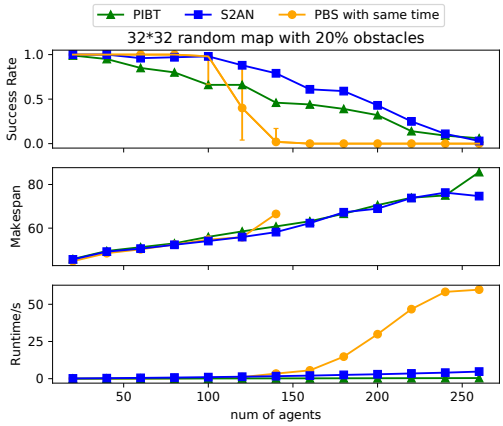


Figure 6: Limited Time Completeness Experiment.

As illustrated in Figure 6, S2AN demonstrates superior performance in terms of success rate when compared to both the same-time PBS and PIBT. Although the average makespan remains nearly identical, S2AN significantly reduces the required time while maintaining a linear time complexity. PBS algorithm employs a tree search approach to enumerate the priority space in search of a feasible solution. In contrast, S2AN generates a specific priority, thereby avoiding the need to search extensively through the priority space. PBS, implemented in C++, exhaustively explores numerous priorities within a similar time frame. In this context, the neural network component of S2AN aids in learning the correct dependencies among agents. For scenarios with a small agent count (less than 100), collisions are infrequent, and PBS operates swiftly. It can quickly iterate through the critical regions of the priority space using conflict-based search, thereby maintaining a 100% success rate in a short timeframe. However, as the agent count surpasses 100, the incidence of collisions increases. In such cases, exhaustive priority space exploration by conflict becomes less efficient and time-consuming. Particularly in scenarios involving 140 to 250 agents, PBS often fails to find any solutions within the allocated time. In contrast, S2AN, guided by a global perspective, swiftly identifies feasible priorities and resolves a substantial portion of the problem just in one-shot. PIBT is a rule-based method, dynamically adjusting priorities at each time step. It is noteworthy that PIBT offers a distributed version that exhibits exceptional speed when fully parallelized. It can be executed in less than 1 second. While effective under specific graph conditions, it tends to struggle and become unsuccessful as obstacle density rises. As in Figure 6, PIBT exhibits a lower success rate compared to S2AN in scenarios characterized by dense obstacles.

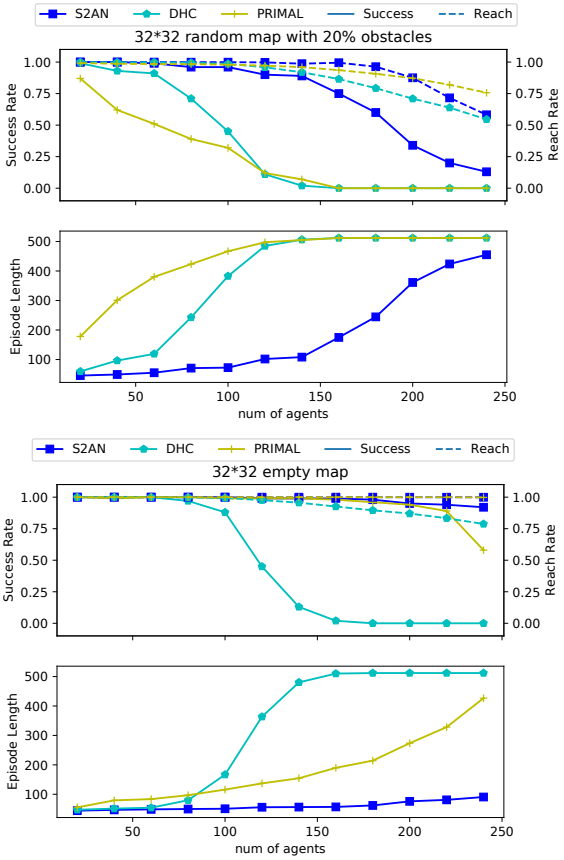


Figure 7: Priority Learning Paradigm Effectiveness. In the double y-axis sub-figure, the solid line represents the success rate, and the dotted line represents the reach rate.

Table 1: Ablation Performance. The values in parentheses represent the percentage relative to the S2AN method.

Method	Test Reward	Converge TimeStep/10^3
S2AN	17.77	45.5
no Entropy	**7.53(-57%↓)**	21.8
No Target Matrix	**7.13(-59%↓)**	16.7
No Harmful	17.12(-4%↓)	**73.0(+60%↑)**
Random	3	None

5.3 Priority Learning Paradigm Effectiveness

To rigorously verify that our framework can achieve better results due to its explicit evasion of multimodal policy learning and inconsistent behavior, we compare the success rate, reach rate, and episode length across various learning methods. Mainstream learning methods adopt a distributed way. Given the current position, field of view (FOV), and communication messages, these methods produce an action at each timestep. By giving a maximum episode length, we can quantify experimental outcomes. Given that the average makespan in non-learning approaches is below 100, we design the maximum episode length as 512. Herein, episode length refers to the average length required to address the problem. The reach rate denotes the fraction of agents who successfully reach their goals by the episode's conclusion. Even if an episode is not deemed successful, learning techniques can still plan a not-feasible solution within a constrained timeframe and achieve a high reach rate. To thoroughly demonstrate the performance of our method, we augmented it with a Reachability Repair Algorithm described in Section 4.5.

As depicted in Figure 7, our performance results surpass those of both PRIMAIL [21] and the communication-based learning method DHC [17]. In scenarios with an empty map, S2AN demonstrates comparable reach rates to PRIMAL. However, S2AN outperforms PRIMAL in terms of success rate when the agent count exceeds 200. DHC leverages strong heuristic map features, but its efficacy diminishes notably in empty map scenarios, resulting in lower success and reach rates. Regarding average episode length, S2AN excels primarily due to its ability to avoid inconsistent behaviors. Notably, as the number of agents increases, the episode length required for problem resolution experiences a significant increment, accompanied by a drop in solution quality. In the 32×32 random map with 20% obstacle density, S2AN consistently maintains an advantage over the other two methods. S2AN exhibits lower reach rates compared to PRIMAL when n exceeds 180. This suggests that the repair method employed may not be robust enough, even though S2AN still outperforms PRIMAL. As discussed previously, the reachability repair method proves valuable in enhancing S2AN's reach rate performance without imposing excessive computational demands. Learning within the priority space appears to offer improved completeness and solution quality, especially in scenarios with a large number of agents. By providing a global map and sacrificing a reasonable amount of time, we can achieve enhanced performance through this learning approach.

5.4 Ablation Study

We conducted an ablation study to evaluate the effectiveness of the Synthetic Score-Based Attention Network (S2AN) and the entropy loss. In order to assess S2AN, we compared it to the original Attention Model [8], with the key difference being the inclusion of domain-specific features, *harmful feature* and the *target matrix*. Therefore, we present results without these two features, and separately without the entropy loss component. Additionally, for this study, the success reward was set to a value of 20, resulting in a maximum achievable reward of 20.

As shown in Table 1, when the entropy loss is omitted, we observe a significant 57% decrease in the test reward. Nevertheless, this performance remains better compared to a random policy, suggesting that the network may quickly converge to a suboptimal policy from which it struggles to escape. Another notable impact on performance arises from the absence of the *target matrix*. This key feature plays a crucial role in guiding the network by providing an approximate feasible solution, effectively indicating which agents should be retained to solve the problem. Its inclusion significantly enhances the network's ability to learn a good policy. Conversely, the *harmful feature* has minimal impact on performance. In practice, the network can acquire local connectivity information through the ResNet, resulting in negligible performance differences. However, the *harmful feature* can accelerate the learning process, reducing the time required to reach the same policy by up to 60% when compared to S2AN.

6 CONCLUSION

In this paper, we propose an effective learning-based paradigm that fuses prioritized planning, which is aimed at addressing the challenge of solving large-scale MAPF in a limited time by fusing the advantages of both classes of methods. By incorporating prioritized planning into the attention-based learning algorithm, we have achieved comprehensive collaborative strategies among multiple agents, elevating both completeness and solution quality within linear time complexity. Simultaneously, by integrating learning mechanisms into the prioritized planning framework and formulating prioritized planning as the Markov Decision Process, the method enables a more informed search process for MAPF. Experiments demonstrate that our attention-based priority learning method achieves a balance between computational time and completeness, outperforming a range of learning-based planners in large-scale pathfinding tasks in terms of success rate, reach rate, and solution quality. Moreover, compared to PIBT and restricted-time PBS, our proposed method exhibits high completeness in medium to large-scale dense obstacle scenarios, making it suitable for expediting various MAPF algorithms.

ACKNOWLEDGMENTS

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant, as well as by an Amazon Research Award. This work was also funded by the National Natural Science Foundation of China, the Science Fund for Creative Research Groups (52221005), and the National Natural Science Foundation of China (Key Project 52131201), as well as the China Scholarship Council.

REFERENCES

[1] Anton Andreychuk, Konstantin Yakovlev, Pavel Surynek, Dor Atzmon, and Roni Stern. 2022. Multi-agent pathfinding with continuous time. *Artificial Intelligence* 305 (April 2022), 103662. https://doi.org/10.1016/j.artint.2022.103662

[2] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. 2014. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. *Proceedings of the International Symposium on Combinatorial Search* 5, 1 (2014), 19–27. https://doi.org/10.1609/socs.v5i1.18315 Number: 1.

[3] Shao-Hung Chan, Roni Stern, Ariel Felner, and Sven Koenig. 2023. Greedy Priority-Based Search for Suboptimal Multi-Agent Path Finding. *Proceedings of the International Symposium on Combinatorial Search* 16, 1 (July 2023), 11–19. https://doi.org/10.1609/socs.v16i1.27278 Number: 1.

[4] Liron Cohen, Tansel Uras, T. K. Kumar, and Sven Koenig. 2019. Optimal and Bounded-Suboptimal Multi-Agent Motion Planning. *Proceedings of the International Symposium on Combinatorial Search* 10, 1 (2019), 44–51. https://doi.org/10.1609/socs.v10i1.18501

[5] Ariel Felner, Roni Stern, Solomon Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. *Proceedings of the International Symposium on Combinatorial Search* 8, 1 (2017), 29–37. https://doi.org/10.1609/socs.v8i1.18423

[6] Omri Kaduri, Eli Boyarski, and Roni Stern. 2020. Algorithm Selection for Optimal Multi-Agent Pathfinding. *Proceedings of the International Conference on Automated Planning and Scheduling* 30 (June 2020), 161–165. https://doi.org/10.1609/icaps.v30i1.6657

[7] Kazumi Kasaura, Mai Nishimura, and Ryo Yonetani. 2022. Prioritized Safe Interval Path Planning for Multi-Agent Pathfinding With Continuous Time on 2D Roadmaps. *IEEE Robotics and Automation Letters* 7, 4 (Oct. 2022), 10494–10501. https://doi.org/10.1109/LRA.2022.3187265 Conference Name: IEEE Robotics and Automation Letters.

[8] Wouter Kool, Herke van Hoof, and Max Welling. 2019. Attention, Learn to Solve Routing Problems!. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ByxBFsRqYm

[9] Sarit Kraus. 1997. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence* 94, 1 (July 1997), 79–97. https://doi.org/10.1016/S0004-3702(97)00025-8

[10] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 21188–21198. https://proceedings.neurips.cc/paper/2020/hash/f231f2107df69eab0a3862d50018a9b2-Abstract.html

[11] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J. Stuckey, and Sven Koenig. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 9 (June 2022), 10256–10265. https://doi.org/10.1609/aaai.v36i9.21266

[12] Jiaoyang Li, The Anh Hoang, Eugene Lin, Hai L. Vu, and Sven Koenig. 2023. Intersection Coordination with Priority-Based Search for Autonomous Vehicles. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 10 (June 2023), 11578–11585. https://doi.org/10.1609/aaai.v37i10.26368

[13] Wenhao Li, Hongjun Chen, Bo Jin, Wenzhe Tan, Hongyuan Zha, and Xiangfeng Wang. 2022. Multi-Agent Path Finding with Prioritized Communication Learning. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 10695–10701. https://doi.org/10.1109/ICRA46639.2022.9811643 Place: Philadelphia, PA, USA tex.eventtitle: 2022 IEEE International Conference on Robotics and Automation (ICRA).

[14] Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. 2019. Searching with Consistent Prioritization for Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 1 (July 2019), 7643–7650. https://doi.org/10.1609/aaai.v33i01.33017643

[15] Hang Ma, Sven Koenig, Nora Ayanian, Liron Cohen, Wolfgang Hoenig, T. K. Satish Kumar, Tansel Uras, Hong Xu, Craig Tovey, and Guni Sharon. 2017. *Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios*. Technical Report. arXiv. https://doi.org/10.48550/arXiv.1702.05515 arXiv: 1702.05515 [cs].

[16] Hang Ma, Jingxing Yang, Liron Cohen, T. K. Kumar, and Sven Koenig. 2017. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 13, 1 (2017), 270–272. https://doi.org/10.1609/aiide.v13i1.12919

[17] Ziyuan Ma, Yudong Luo, and Hang Ma. 2021. Distributed Heuristic Multi-Agent Path Finding with Communication. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 8699–8705. https://doi.org/10.1109/ICRA48506.2021.9560748 ISSN: 2577-087X.

[18] Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence* 310 (Sept. 2022), 103752. https://doi.org/10.1016/j.artint.2022.103752

[19] Jingyao Ren, Vikraman Sathiyanarayanan, Eric Ewing, Baskin Senbaslar, and Nora Ayanian. 2021. MAPFAST: A Deep Algorithm Selector for Multi Agent Path Finding using Shortest Path Embeddings. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1055–1063.

[20] Qandeel Sajid, Ryan Luna, and Kostas Bekris. 2012. Multi-Agent Pathfinding with Simultaneous Execution of Single-Agent Primitives. *Proceedings of the International Symposium on Combinatorial Search* 3, 1 (2012), 88–96. https://doi.org/10.1609/socs.v3i1.18243 Number: 1.

[21] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters* 4, 3 (July 2019), 2378–2385. https://doi.org/10.1109/LRA.2019.2903261

[22] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219 (Feb. 2015), 40–66. https://doi.org/10.1016/j.artint.2014.11.006

[23] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Kumar, Roman Barták, and Eli Boyarski. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Proceedings of the International Symposium on Combinatorial Search* 10, 1 (2019), 151–158. https://doi.org/10.1609/socs.v10i1.18510

[24] Pavel Surynek. 2020. Bounded Sub-optimal Multi-Robot Path Planning Using Satisfiability Modulo Theory (SMT) Approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 11631–11637. https://doi.org/10.1109/IROS45743.2020.9341047 ISSN: 2153-0866.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[26] Hanlin Wang and Michael Rubenstein. 2020. Walk, Stop, Count, and Swap: Decentralized Multi-Agent Path Finding With Theoretical Guarantees. *IEEE Robotics and Automation Letters* 5, 2 (April 2020), 1119–1126. https://doi.org/10.1109/LRA.2020.2967317 Conference Name: IEEE Robotics and Automation Letters.

[27] Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. 2023. SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding. https://doi.org/10.48550/arXiv.2303.00605

[28] Jens Weise, Sebastian Mai, Heiner Zille, and Sanaz Mostaghim. 2020. On the Scalable Multi-Objective Multi-Agent Pathfinding Problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8. https://doi.org/10.1109/CEC48606.2020.9185585 Place: Glasgow, United Kingdom tex.eventtitle: 2020 IEEE Congress on Evolutionary Computation (CEC).

[29] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3 (May 1992), 229–256. https://doi.org/10.1007/BF00992696