POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

An empirical comparison of multi-agent path finding algorithms: classic and learning-based models in realistic warehouse environments

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Andrea Giuffrida

Advisor: Prof. Francesco Amigoni

Co-advisor: Prof. Nicola Basilico

Academic year: 2022-2023

---

1. Introduction

Multi-agent pathfinding (MAPF) [1, 2] is a fundamental challenge within robotics and artificial intelligence. It involves the task of planning and executing collision-free paths for multiple agents operating within a shared environment. MAPF has many applications, like warehouse automation and logistics autonomous vehicle navigation, game design, and robotics. The classic MAPF approach typically relies on centralized algorithms, which assume complete knowledge of the environment and agent states. However, MAPF is an NP-hard problem. This complexity limits the effectiveness of centralized approaches for large teams, impacting solution quality and scalability. Recent advancements in Deep Reinforcement Learning (DRL) [3] and Multi-Agent Reinforcement Learning (MARL) [4], offer an interesting alternative to traditional approaches for MAPF. These novel approaches enable the execution of the algorithms to become decentralized, relieving the computational cost of a single central computational unit. However, learning-based models often require extensive training, and they do not provide any kind of guarantees on the solution.

The central focus of this thesis is an extensive comparison of centralized and learning-based MAPF algorithms in realistic environments. The models used are AB-MAPPER [5], DCC [6], PRIMAL [7], SCRIMP [8], CBS [9], CBSH2-RTC [10], and ODrM* [11]. We investigate whether performance differences exist between centralized and learning-based methods in such environments, if specific algorithms excel in handling particular challenges, and to what extent learning-based approaches can match the solution quality and computational efficiency of centralized algorithms. In particular, we address three specific aspects: the environment design, the algorithms' adaptation to function effectively within the designed environments, and the comparative empirical analysis.

2. Related Work

2.1. MAPF problem formulation

The most common MAPF mathematical representation involves a graph $G = (V, E)$, where V is the set of vertices that represent all the available positions in the environment, and E is the set of edges connecting the vertices of the graph. We also have n agents $A = \{a_1, \ldots, a_n\}$ where each agent a_i has a unique start vertex $(s_i \in V)$

and a unique goal vertex ($g_i \in V$). The time is discretized and at each timestep each agent can either move to one of the adjacent vertices in the graph through an edge or stay in the current vertex. Two kinds of collisions can happen: two agents are in the same vertex at the same timestep, causing a vertex collision, or two agents tries to move along the same edge at the same timestep, causing an edge collision. The objective of MAPF is to find a collision-free path for each agent, where the first vertex of the path is the starting location s_i and the last vertex is the goal g_i. MAPF can be optimized with different objectives, where the most common ones are the makespan and the sum-of-costs.

Typically this MAPF formulation translates into an implementation of a 2D discrete 4-connected grid environment, with 5 valid actions: up, right, left, down, and stay.

2.2. Centralized Algorithms

Centralized algorithms are the oldest and classic type of MAPF solvers. They monitor the overall system state and create individual plans for each agent. Centralized MAPF solvers can be essentially divided into three subcategories:

- **Reduction-based**. The core of these algorithms lies in their ability to transform the MAPF problem into well-established combinatorial optimization problems. This approach ensures completeness and optimality (usually only for the makespan). They are typically used in small-sized graphs with densely placed agents.
- **Rule-based**. These algorithms use a set of primitive operations that specify the actions of the agents in different situations. They often guarantee complete solutions and they do not guarantee optimality, but they are very efficient, using just a simple rule.
- **Search-based**. These algorithms leverage heuristic and search techniques, and they have been the focal point of MAPF research in the past decade. Some of the most famous MAPF algorithms fall into this category, like M*, Conflict Based Search (CBS), and Increasing Cost Tree Search (ICTS). Usually, these algorithms use a 2-level hierarchical structure: the high level checks the solutions computed at the low level and imposes constraints on the paths. This part of the algorithm is typically centralized. Then, the low level searches for possible paths with the constrained imposed by the high level, and passes them to the level above.

Unfortunately, all of these solvers have a problem: it is well known that MAPF is an NP-hard problem and so very difficult to solve optimally. In particular, as the number of agents or the environment size increases, the computation time required increases, up to the point that these kinds of solvers become infeasible to use.

2.3. Decentralized and Learning Models

To address limitations in centralized planning (see Section 2.2), research has shifted towards decentralized approaches. Instead of a single algorithm planning for all agents, each agent is given "local intelligence" to complete its task and collaborate with others. This intelligence often comes from Deep Reinforcement Learning (DRL) algorithms, leading to the emerging field of Multi-Agent Reinforcement Learning (MARL).

In MARL algorithms, agents learn how to reach goals, avoid obstacles, and cooperate with others through trial and error, receiving rewards for successful actions. Some algorithms also incorporate Imitation Learning (IL), where agents learn by observing expert behavior.

Learning-based models can be classified using several factors:

- **Training method**. Training is a key part of every framework based on learning. Training can be completely centralized, with a central controller that receives all agent's observations, update the parameters, and send the computed actions back to the agents. Then we can also have independent learners, where each agent possesses and manages its own value function or policy, treating all the other agents as part of the environment. But the most used method is Centralized Training and Decentralized Execution (CTDE). It presents a hybrid approach to MARL, combining both centralized and decentralized processing. During the training phase, agents have access to more information via a central entity, while the execution phase remains decentralized, with each agent basing its actions only on its local observations.

- **DRL algorithm**. In a learning-based model, the underlying DRL algorithm is crucial. Essentially DRL algorithms are divided into three categories. The first category is the value-based methods that tries to approximate and learn the optimal value function. Then, the actions are simply selected using the function learned. Following this approach, the value function is either decomposed into smaller, simpler parts or it is approximated by a neural network. The second type is the policy-based methods, which directly approximates the policy that select the actions. Finally, we have the actor-critic methods, which combine value and policy based methods. It employs two neural networks: the actor network which maps local observations to actions, approximating the policy, and the critic network which evaluates the actions performed, approximating the value function. Each network uses the other to update itself.
- **Other factors**. The models can be further characterized by other factors. For example, they could rely solely on local information, they could employ inter-agent communication, or they could use a global guidance algorithm like A*.
  Moreover, they could use attention in different parts of the model or employ the curriculum learning technique, which will present to the model training cases of increasing difficulties.

3. Experimental Setting

This thesis has the objective of performing a comprehensive experimental comparison of some DRL-based algorithms within realistic warehouse environments, using nine different metrics. We want to test how these algorithms perform out-of-the-box, without any particular type of retraining or fine-tuning. Moreover, a focal point of the work is to compare them with some classic centralized algorithms.

3.1. Algorithms Description

In this section we will describe the details of the learning-based algorithms used.

PRIMAL [7] introduced the application of MARL to MAPF challenges. It combines RL and IL (Imitation Learning). Agents possess a restricted field of view (FOV) of 10×10 cells centered on their position. Each agent in PRIMAL has 4 input channels of the same dimension of the FOV. These contain the obstacle positions, the other agent's location, the neighbors' goals, and the agent's goal. Additionally, each agent is provided with a goal-direction vector and the Euclidean distance to its target. To improve training, PRIMAL uses two key strategies: an auxiliary loss for blocking penalties and an auxiliary loss for the IL part. It uses the A3C algorithm to train the networks and uses the rewards shown in Table 1 to guide the algorithm.

Action	Reward
Move (Up/Right/Left/Down)	-0.3
Agent Collision	-2.0
No Movement (on/off goal)	0.0 / -0.5
Finish Episode	20.0

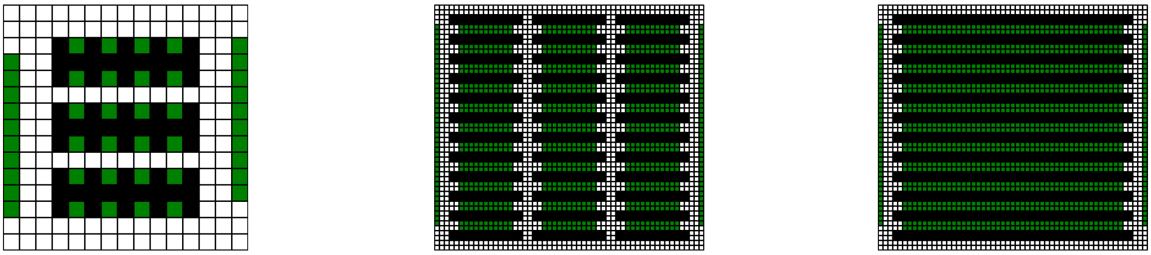
Table 1: PRIMAL reward structure.

The second model we used is AB-MAPPER [5], designed for MAPF with dynamic obstacles, that introduced three critical enhancements: an actor network leveraging BicNet (bi-directional LSTM communication) for robust inter-agent communication, a critic network integrating an attention mechanism to selectively evaluate actor network performance, and a global A* path to improve the navigation capabilities.

Reward	Value
Step penalty r_s	-0.1, move / -0.5, wait
Collision penalty r_c	-5
Oscillation penalty r_o	-0.3
Off-route penalty r_f	$-d_{min}$
Goal-reaching reward r_g	30

Table 2: AB-MAPPER reward structure.

It also adopts the actor-critic architecture but the rewards used (Table 2) are slightly differ-



(a) 15 × 15 environment. (b) 50 × 55 environment. (c) 50 × 55 “long shelves” environment.

Figure 1: Realistic warehouse environments.

ent from the PRIMAL ones: it introduced the oscillation penalty, which punishes the oscillatory movements, and the off-route penalties, where d_{min} is the minimum distance between the agent’s current position and the A* path.

DCC [6] focuses on the communication part. Previous models used broadcast communication, which could lead to bandwidth saturation, latency, and redundant information. Improving the previous models, DCC implements a request-reply algorithm, where an agent j is sent a request from agent i only if the presence of j is enough to alter the policy of i. The communication is then managed by a block that performs a graph convolution with multi-head dot-production as the convolutional kernel. DCC also uses some heuristics in the input channels, that will help in the agents’ navigation. It uses a dueling Q-network to approximate the value function and select the action. During training, DCC makes use of the Ape-X framework, which employs a combination of experience parallelization and prioritized experience replay. Essentially, it has a single learner on a GPU, that updates the experience priority and the network parameters, and multiple actors on the CPU, which will generate the experience to store in the shared memory following the policy based on the value function produced by the network. The rewards are the same as PRIMAL, but changes in the values.

The last model we used is SCRIMP [8]. With large teams, agents get overloaded with conflicting messages, creating the so-called “chatter problem”. SCRIMP addresses it by using a limited smaller FOV (3 × 3) and highly scalable global communication using transformer-based attention mechanisms. Other key features of SCRIMP include a learning-based stochastic tiebreaking method to manage conflicts and an additional reward mechanism to improve the single-agent exploration, punishing agents that wander around the same area instead of exploring. SCRIMP is trained with the Centralized Training and Decentralized Execution method (Section 2.3), using Proximal Policy Optimization (PPO) as DRL algorithm. Also SCRIMP’s rewards, as DCC, are inspired by the PRIMAL ones.

3.2. Maps

All the models described above, except for AB-MAPPER, were trained and tested into random environments. While random maps can be good for an initial benchmark, they do not necessarily hint at the performance the models will have if deployed into a map with some realistic purpose. Therefore we decided to test the performance of these models in a warehouse-like environment, to understand their behavior and how they cope with realistic challenges. The environments we used are three:

- The first is the smaller environment of size 15×15 in Figure 1a. Not many agents could fit in, but it is a good example to study the algorithms in crowded and dense situations.
- The second environment has size 50×55 (Figure 1b). This environment is able to cope with more agents navigating inside and, therefore is a good benchmark for long-distance goals in a less crowded environment, or large group agent management.
- The last environment, shown in Figure 1c, is a copy of the second environment but with some open areas between the “shelves”

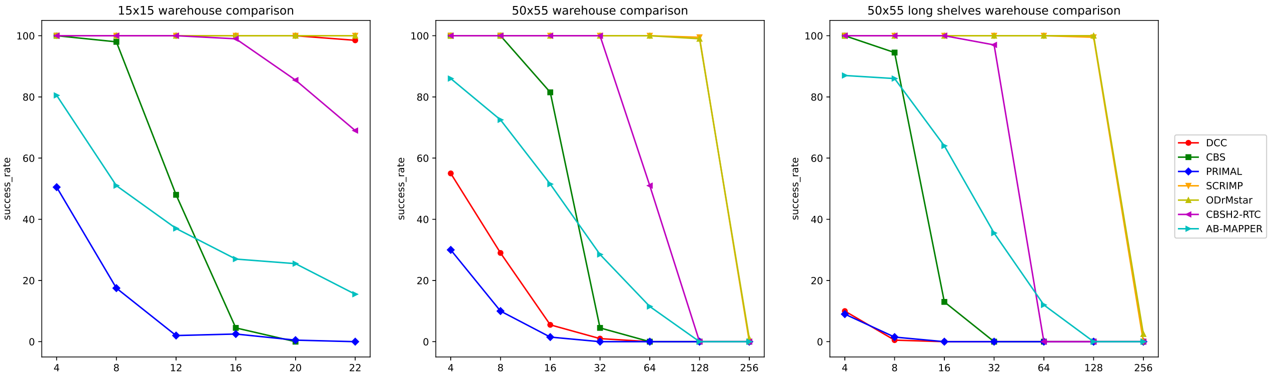


Figure 2: Success rate of the models vs. number of agents.

closed. This might push the cooperation evaluation even further, being the hardest environment for large agent groups.

In the environment images, some cells are colored in green. These are the cells belonging to the *open list*, that represents the set from which the start and goal cells can be selected for each run.

4. Results

4.1. Evaluation Metrics

We used nine metrics to evaluate the performance of the models in our environments:

- The success rate, that measures the percentage of completed episodes.
- The execution time.
- The episode length, which is the length of the episode in discrete time steps.
- The total, average, and maximum distance across the agents in a single episode.
- The collision rate, which measures the collisions detected during an episode, and is further divided into obstacle collision rate and agent collision rate.

4.2. Results Discussion

We must consider that each algorithm has a limit on the episode length: if during the episode this limit is crossed, then the episode is declared unsuccessful. PRIMAL, DCC, and SCRIMP have this limit set to 256 steps. In AB-MAPPER it is dynamic and depends on the longest single-agent A* path and a parameter, which essentially translates into longer episodes. Beyond the learning models described in Section 2.3, we used three centralized algorithms (CBS, CBSH2-RTC, and ODrM*), optimal on the sum-of-costs. In Figure 2 we can observe the success rate comparison of the models across different environments. PRIMAL is definitely the worst in terms of success rate in all the environments, probably because of the lack of agents' coordination, but it is the only model to have 0 collisions in all the environments. DCC is very good in the smaller environment, where it is able to solve almost all the episodes. In 50×55 sized environments it struggles a lot more, reaching the same performance of PRIMAL in the hardest environment. We can note in Table 3 how DCC's episode length is almost at the limit of 256 steps, explaining the few successes and showing the poor quality of the paths produced. It showed good agent cooperation, due to the communication module, but poor navigation performance. AB-MAPPER has a success rate that degrades almost linearly. Although it appears to have a good success rate, we need to take into account that its maximum episode length is longer than the other algorithms, resulting in higher success rates but longer paths. Most likely, many agents wander around before eventually reaching their goals, which increase the success rate but comes at the expense of longer paths, as illustrated in Table 3. SCRIMP basically has always 100% success rate, struggling only with 256 agents. It showed good capabilities both in coordination and navigation, excelling both in small crowded environments and large maps. It is comparable in terms of paths' quality with the centralized algorithms, as Table 3 shows. Among centralized algorithms, CBS does not scale well in settings with more agents, (up to 16 agents in the first environment, 32 in the second and third environment), and the execution time is much longer compared to the other algorithms, partially justified because it is implemented in Python.

Model	Episode length
PRIMAL	199.33
DCC	255.09
AB-MAPPER	666.31
SCRIMP	71.365
CBS	70.67
CBSH2-RTC	70.76
ODrM*	70.94

Table 3: Comparison of episode length in 50 × 55 long shelves environment with 8 agents.

CBSH2-RTC works well, having a 100% the success rate for fewer agents, but decreasing in harder scenarios. Finally, ODrM* is probably still the best overall and it is the only one able to solve, in a reasonable time, a few cases with 256 agents.

Notably, the centralized algorithms have similar but different episode lengths because they are optimized on the sum-of-costs and not the episode length.

5. Conclusions

This study aimed to provide a comprehensive comparison of learning-based models applied to the MAPF problem, within more realistic settings inspired by warehouse logistics. We designed three distinct maps, each with unique characteristics, to evaluate the adaptability of various models. We meticulously adapted each algorithm's implementation to work within our custom environments, ensuring consistent output formats for in-depth comparative analysis. A suite of nine performance metrics was utilized. This evaluation framework enabled us to dissect algorithmic strengths and weaknesses. The experimental results showed how DCC (only in the smaller environment) and SCRIMP (also in larger ones) can almost match the performance of centralized algorithms. This finding hints at the potential of DRL-based approaches for real-world MAPF problem-solving.

There are several promising avenues for future investigation, that could be inspired by this work: the evaluation of new models or some hybrid centralized and learning-based approaches, an analysis of the impact of the single model components, an environment-specific fine-tuning of the models, and the implementation of a physical robot simulation.

References

[1] H. Ma and S. Koenig, "Ai buzzwords explained: multi-agent path finding (mapf)," *AI Matters*, vol. 3, no. 3, pp. 15–19, 2017.

[2] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. SoSC*, vol. 10, pp. 151–158, 2019.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[4] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. ICML*, pp. 330–337, 1993.

[5] H. Guan, Y. Gao, M. Zhao, Y. Yang, F. Deng, and T. L. Lam, "Ab-mapper: Attention and bicnet based multi-agent path planning for dynamic environment," in *Proc. IROS*, pp. 13799–13806, IEEE, 2022.

[6] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1455–1462, 2021.

[7] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[8] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "Scrimp: Scalable communication

for reinforcement-and imitation-learning-based multi-agent pathfinding," in *Proc. IROS*, pp. 9301–9308, IEEE, 2023.

[9] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[10] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for multi-agent path finding with conflict-based search.," in *Proc. IJCAI*, vol. 2019, pp. 442–449, 2019.

[11] C. Ferner, G. Wagner, and H. Choset, "Odrm* optimal multirobot path planning in low dimensional search spaces," in *Proc. ICRA*, pp. 3854–3859, IEEE, 2013.