2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Conflict Area Prediction for Boosting Search-Based Multi-Agent Pathfinding Algorithms

Jaesung Ryu[1], Youngjoon Kwon[1], Sangho Yoon[1], and *Kyungjae Lee[1]

***Abstract*— We address the challenge of efficiently controlling multi-agent systems, crucial in fields like logistics and traffic management. We propose a novel approach that combines learning-based techniques with search-based methods, focusing on enhancing the conflict-based search (CBS). The CBS ensures optimality but suffers from increasing complexity as agents or maps grow. To tackle this, we leverage learning-based approaches to enhance computational efficiency. By training a conflict area prediction (CAP) network, we anticipate potential conflict areas, allowing for low-level path planners to explore conflict-free paths. Our experiments demonstrate the effectiveness of our method in reducing computational demands compared to existing approaches.**

I. Introduction

Recently, multi-agent control systems have found extensive applications across various industries. They are utilized for controlling multiple vehicles and mobile robots [1] in scenarios such as large-scale logistics warehouses [2], traffic management [3], and congested airports [4]. Finding conflict-free and shortest paths for multiple agents is essential to efficiently control these multi-agent systems. Multi-agent path finding (MAPF) is a problem in which multiple agents, each with their own start and goal locations, need to find optimal paths to their respective goals while avoiding conflicts with each other. To solve the MAPF problem, various search-based methods [5]–[10] have been developed to ensure optimality and completeness, like a conflict-based search (CBS) [11]. However, there still remain significant challenges as the number of agents increases or the map size grows, resulting in exponentially increasing computational complexity.

From the perspective of computational complexity, learning-based methods have recently been developed to reduce computational loads. Learning-based methods train path planners (or controllers) for each agent to reach their target locations without conflicts with other agents by using imitation learning [12], or guiding agents with their shortest path information, or enhancing cooperative behaviors [13]. While the training phase requires significant computational resources, once the training is completed, it can find paths with several inferences of the trained model.

*:Corresponding author.
[1]J.Ryu, Y.Kwon, S.Yoon, K.Lee are with the Department of Artificial Intelligence, Chung-Ang University, Seoul 06974, Republic of Korea (e-mails: jaesungryu96@gmail.com, dud842655@gmail.com, sh.y970719@gmail.com, kyungjae.lee@ai.cau.ac.kr)

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea(NRF) grant funded by the Korean government (MSIT) (No.2021-0-01341, AI Graduate School Program, CAU, 50%) and (No. RS-2023-00211357, Smart Assembler: Robot Active Learning for Unseen Parts Assembly, 50%), respectively.

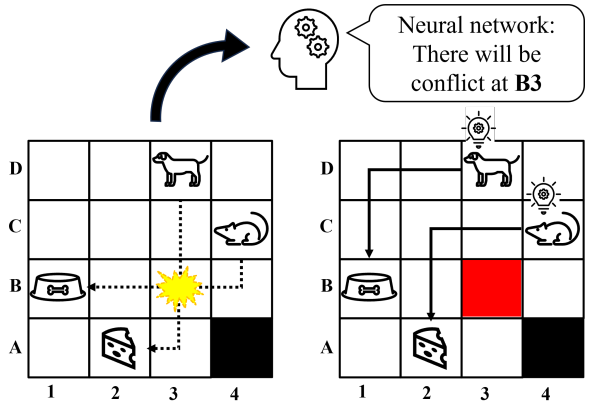


Fig. 1: Motivation to reduce conflicts in advance. A dog and mouse aim to reach $B1$ and $A2$, respectively, but their initial plans have the conflict at $B3$. If a cost is added at $B3$, their initial plan can be changed to conflict-free ones.

While the majority of learning-based methods efficiently reduce computational time compared to search-based methods, they do not guarantee optimality and completeness, which are critical for industrial applications. In this regard, we propose a novel approach that leverages learning to boost the computational efficiency of search-based methods while preserving these properties. We mainly improve the CBS which is known as an effective optimal MAPF solver. CBS typically operates on two levels. The low-level focuses on finding the optimal paths for individual agents. When these paths encounter conflicts, the high-level enforces constraints on the conflicting agents to prevent these conflicts, where optimal is shortest. Search-based methods are applied at the high-level process, exploring tree searches to determine what constraints need to be added to resolve conflicts while finding an optimal path. In this regard, as the number of conflicts increases, the computational demand and memory requirements increase. To address this, we train a conflict area prediction network to anticipate the areas where conflicts may arise. Using these predictions, we can prioritize the exploration of non-conflict paths in the low-level process as shown in Fig. 1. To validate our method, we conducted experiments on various MAPF scenarios with various numbers of agents and different map sizes, comparing the proposed method with various search-based and learning-based MAPF algorithms. In experiments, we confirmed that the proposed method is approximately 9.3% better than learning-based approaches in terms of optimality and around 40% faster than search-based methods in terms of computation time.

II. Related Work

Algorithms for solving the MAPF problem are continuously being researched. Among these, a CBS [11] is well-known for its conflict-free and optimal characteristics. CBS encounters a challenge such as exponentially increasing computation time as the number of agents or scenario complexity grows, due to the increasing conflicts that need to be resolved in the constraint tree. Consequently, various approaches have been proposed to address these issues.

Some approaches to the CBS algorithm [11] involve adding specific methods to reduce the size of constraint tree [14] or to sacrifice optimality to decrease computation time [15], [16]. Improved-CBS (ICBS) [14] utilizes prioritizing conflicts (PC) and bypass (BP) techniques within the CBS algorithm. It reduces the constraint tree size by validating newly discovered paths before expanding nodes. The validity of the path means to resolve conflicts through constraints. Enhanced-CBS (ECBS) [15] sacrifices optimality to some extent but introduces a method to find bounded sub-optimal paths within limited compromise, demonstrating the ability to discover paths in a shorter time, even in cases where CBS struggles to find a solution within a time limit. The bound size can be adjusted by a parameter called w, with $w = 1$ corresponding to standard CBS. Hierarchical cooperative A* (HCA*) [16] is another sub-optimal search-based algorithm. It finds the shortest path for one agent, and this path occupies the global reservation table. When the next agent searches for a path, the areas occupied by the previous agents are considered as obstacles, and the path is found with those areas already blocked.

There are also approaches to solving MAPF problems using learning techniques [12], [13], [17], [18]. In [12], a path finding via reinforcement And imitation multi-agent learning (PRIMAL) has been proposed where PRIMAL addresses the computational challenges of the existing CBS-like algorithms. PRIMAL is inspired by the concept of enabling each agent to navigate independently, avoid conflicts, and reach their targets to quickly solve the MAPF problem. In [17], a globally guided reinforcement learning (G2RL) has been proposed where the ratio of detour actions by PRIMAL agents is reduced, such as moving in directions other than the goal direction or getting stuck in a specific area due to obstacles. G2RL informs an agent of one of the possible optimal paths obtained through the A* algorithm and introduces a reward design that penalizes deviations from this path. Furthermore, a distributed heuristic MAPF with communication(DHC) [13] modified G2RL's global guidance by addressing the potential confusion caused by providing a single optimal path as guidance because the optimal path may not be unique. Instead, they changed their approach to provide guidance based on heuristic map information in the direction the agents were moving. Both G2RL and DHC have mainly employed the results of A* which provides optimal and complete examples, however, the resulting policy cannot strongly guarantee optimality and completeness since it just guides the learning processes. SACHA [18] improves cooperative action among agents by supplementing their observations with heuristic map information from neighboring agents. Additionally, it enhances performance by implementing the soft actor-critic framework and proposing an optional communication module.

We propose a novel approach that distinguishes itself by integrating learning techniques with search-based algorithms, specifically designed to boost the search-based algorithm. This integration maintains optimality and completeness while significantly reducing computation time by combining the strengths of both learning and search-based methods.

III. Backgrounds

Given an undirected and connected graph $G = (V, E)$ and N agents indexed by $i = \{1, 2, ..., N\}$, each agent has a unique start vertex $s_i \in V$ and a unique goal vertex $g_i \in V$. Time is represented by discrete time steps $t = 0, 1, ..., \infty$. At each time step, an agent can either wait at the current vertex or move to an adjacent vertex. conflicts between agents involve vertex conflicts when two agents occupy the same vertex simultaneously, and edge conflicts when two agents traverse the same edge in opposite directions. A MAPF solution consists of conflict-free paths for all agents. Commonly used metrics for path quality involve the total sum of path lengths, where the path length indicates the time an agent takes to reach the goal vertex. Our research is conducted on a 4-connected grid where five possible movements are available: left, right, down, up, and wait.

CBS iterates through two processes, the low-level and high-level processes, to find optimal paths as illustrated in Fig. 2. The low-level process employs the A* algorithm to find paths for all agents individually. A* algorithm [19] calculates costs for all vertices in the scenario using the equation $f(v) = g(v) + h(v)$. Here, $g(v)$ represents the distance from the start vertex s_i to the current vertex v, and $h(v)$ is the estimated cost calculated by a specific heuristic function from the current vertex v to the goal vertex g_i. Euclidean and Manhattan distances are commonly used for the heuristic function. In the high-level process, by comparing paths for all agents, detect conflicts, and constraints are added to restrict specific agent movements and resolve conflicts. Adding constraints prompts the low-level process to generate new paths that adhere to the constraints. The resulting tree structure is known as the constraint tree. Once the shortest paths resolving all conflicts are found, the algorithm terminates. ICBS aimed to reduce computation time by effectively reducing the size of the constraint tree.

IV. Methods

A. Conflict Area Prediction

We propose a method to reduce computation time compared to the standard CBS algorithm by introducing an additional heuristic value into the low-level A* search. As previously mentioned, A* calculates estimated costs using the formula $f(v) = g(v) + h(v)$ and finds paths in the direction with the lowest estimated cost. Our approach aims

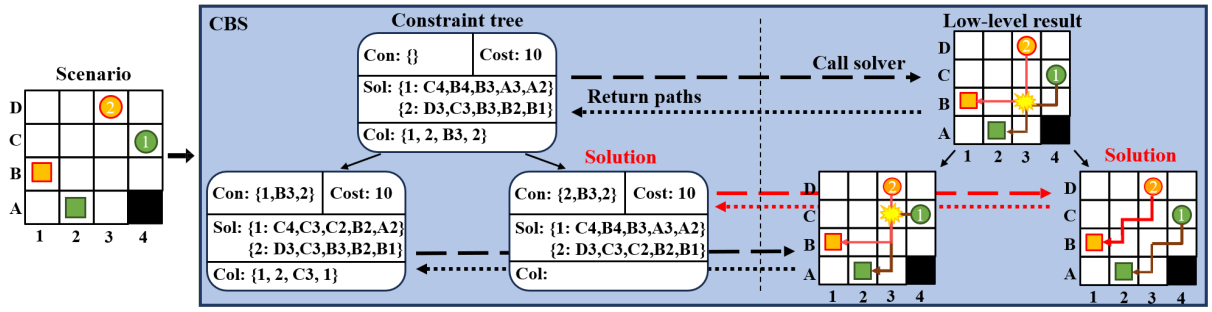


Fig. 2: Illustration of CBS. Given an initial scenario, the lower-level process first computes the shortest path for each agent. CBS generates a root node containing conflict information and paths, and then extends the constraint tree by adding child nodes. These child nodes add a constraint from conflicts in the parent node and compute the new paths that resolve the conflicts. The constraint tree continues to expand until a solution with no conflicts is found

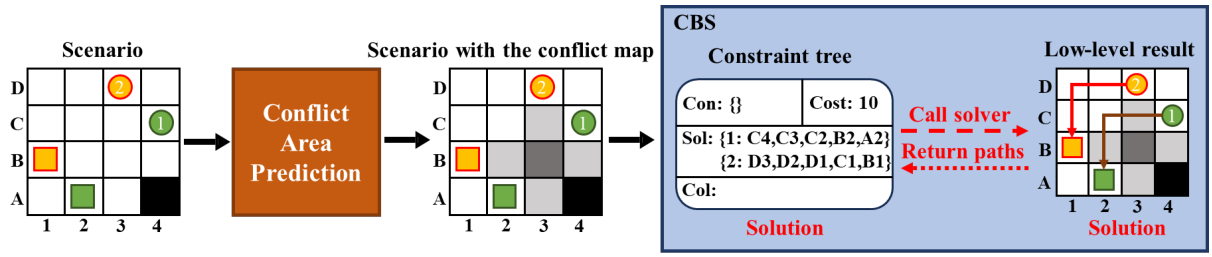


Fig. 3: Illustration of the proposed method. Compared to Fig. 2, the conflict map influences the initial low-level results to prioritize paths that avoid regions where conflicts are likely to occur.

to add an additional heuristic value $cf(v)$ to set the estimated cost of conflict-prone vertices, adjusting the heuristic function to $h'(v) = h(v) + cf(v)$ and consequently, the cost function to $f(v) = g(v) + h'(v)$.This adjustment encourages the A* algorithm to output paths that avoid these conflict-prone vertices.

We use a neural network to predict these conflict areas and generate a conflict map with additional heuristic values in those areas. The predicted conflict map is then provided to CBS, and the subsequent steps are the same as standard CBS. Fig. 3 illustrates cases where our proposed method successfully provides an advantage over traditional CBS.

Additional heuristics generated from the network can alter the solutions of A*. If the network accurately predicts conflict areas, A* will bypass these areas and find the precise optimal solution. However, it is essential to discuss whether the optimality of CBS is compromised in case of incorrect predictions. Fortunately, even if the network mispredicts the conflict areas, CBS can still find the optimal solution while computational demands may increase. We categorize the misprediction scenario into three cases based on an optimal solution: straight scenario, multiple diagonal scenario, and unique diagonal scenario.

1) Straight Scenario: The first scenario involves a goal point positioned horizontally or vertically, resulting in a straight optimal path. When a mispredicted conflict map is added to this path, A* would only deviate from the straight path if the additional heuristic value exceeded 2 since the minimum distance between grids is 1 and the deviation from and recovering back to the straight path requires double transitions. Hence, if we set the range of the conflict map to $cf(v) \in (0, 1)$, then, the misprediction does not hamper A*.

2) Multiple Diagonal Scenario: The second scenario features a goal point positioned diagonally, leading to multiple potential optimal paths. Since there are multiple optimal paths, even if one optimal path is blocked by misprediction, the solver can take an alternative optimal one.

3) Unique Diagonal Scenario: The third scenario is the worst case, which also has a diagonal goal point, but only a unique optimal path is possible due to conflicts with other agents. When a mispredicted conflict map directs the algorithm towards the conflict-prone path, additional time is required for the high-level process to recognize that the path encounters a conflict while not affecting optimality.

In all these cases, we verified that the mispredicted conflict map does not compromise the optimality of CBS. We also experimentally confirmed this property for 22,000 scenarios.

B. Training Data Generation

We employed scenarios from the MAPF benchmark [20]. The positions of obstacles in benchmark scenarios were kept fixed, while the agents' starting positions and goals were randomly placed to generate various scenarios. We selected scenarios solvable by CBS and ICBS within less than five minutes. Simultaneously, we collected conflict data for each scenario from the CBS results. Using this data, we generated the target conflict map for each agent. Locations of conflict data were assigned a cost of 1 and then passed through a Gaussian filter to create a heatmap, which was normalized to values between 0 and 1.The conflict map was utilized as training data for the neural network to predict conflict areas. The input of a neural network consists of map and positional information. Map information consists of obstacle maps and heuristic maps. The obstacle map is encoded such that empty

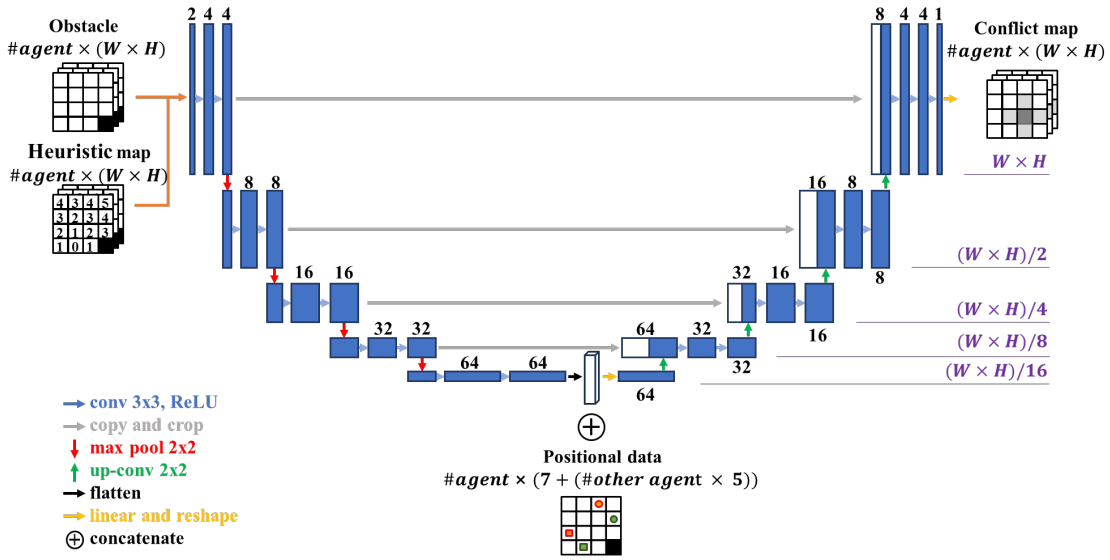


Fig. 4: The structure of the proposed conflict area prediction network.

spaces are represented as 0, while obstacles are represented as 1. The heuristic map's heuristic values are calculated using Dijkstra's algorithm. Positional information includes current agent position (sx, sy), goal point (gx, gy), and goal vector $(\frac{gx-sx}{mag}, \frac{gy-sy}{mag})$, and mag, where mag denotes the distance from the position to the goal point. Positional information also includes the information of other agents, such as the relative position to the current agent $(sx_{other} - sx, sy_{other} - sy)$ and goal vector and mag.

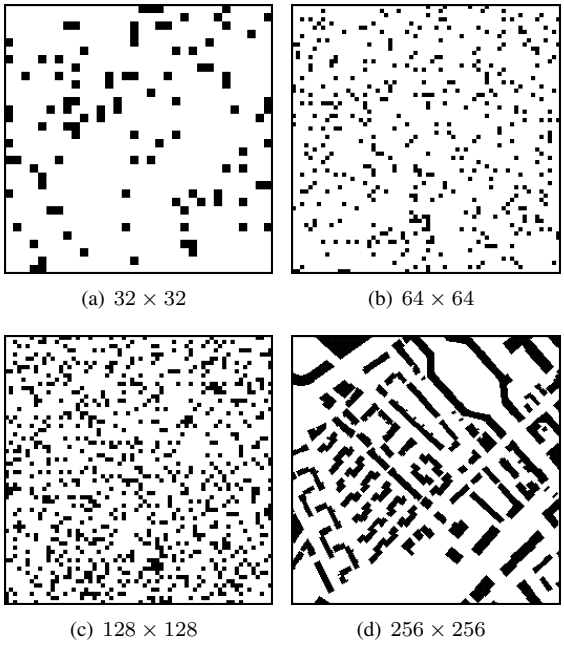
C. Architecture and Training

We constructed a network to predict conflict-prone areas and generate the conflict map. Fig. 4 displays the structure of our network and how the two input data types, map information and positional information, are input into the network. The map information passes through convolution layers and max-pooling layers iteratively until it reaches the bridge. At this point, it is concatenated with the other input data, the positional information. The concatenated data then goes through a linear layer to ensure the sizes match. Afterward, it passes through up-convolution layers and convolution layers iteratively. Finally, it goes through a linear layer to adjust the output size and generate the conflict map. The network takes inputs of dimensions $(\#Agents \times ((2 \times W \times H) + (7 + \#Other\ agents \times 5)))$ and produces outputs of dimensions $(\#Agents \times (W \times H))$. Learning occurs through comparison with the target conflict map generated from conflict data. The mean squared error (MSE) loss function was employed in this process.

V. Experiments

To verify the effectiveness of the proposed method, we compare our algorithm with other MAPF algorithms in various benchmark scenarios. All experiments were tested on a 12th Gen Intel® Core™ i7-12700KF processor and a GeForce RTX 3080 Lite Hash Rate GPU.

We generate data using scenarios from the MAPF benchmark. We utilized scenarios of sizes 32×32, 64×64, and 256×256 (Boston) and additionally scaled up the 64×64 scenario to create a 128×128 scenario, resulting in a total of four scenarios. The number of agents was set to 40 for the 256×256 scenario and 20 for the rest, with agents randomly placed in each scenario. We prepared approximately 11,400 data for the 32×32 scenario and around 3,000 data for each of the other scenarios. These data were divided into a train-validation-test split of $80 : 10 : 10$. The hyperparameters were set as follows: 200 epochs, a learning rate of $1e^{-4}$, Adam optimizer, and a batch size of 32 for the 256×256 scenario and 20 for the others.



(a) 32×32 (b) 64×64

(c) 128×128 (d) 256×256

Fig. 5: Pictures of each scenario.

As our proposed method is built upon CBS, we included CBS for comparison to determine if it could offer faster computation time. Additionally, we incorporated ICBS as a

baseline since it claims enhancements in computation time.

As baselines for learning-based methods, we utilized PRIMAL [12] and DHC [13]. To assess whether our proposed method can achieve a similar computation time to learning-based methods, we included these baselines. Furthermore, learning-based methods do not guarantee optimality. We also compared how our proposed approach differs in terms of optimality and success rate from those methods.

We evaluated performance using several metrics to compare our proposed approach with baseline methods. To verify the computational efficiency, we first assessed the size of constraint trees for CBS, ICBS, and ours by measuring the number of generated nodes n_{Gen} and the number of expanded nodes n_{Exp}. Furthermore, total CPU time taken to solve the scenario is also measure where computation time is calculated from scenario input to solution output, excluding environment setup time for learning-based methods. Success rate measures the ability to complete a task within the given time steps. The experiment was conducted with scenarios that were successful in both CBS and ICBS to verify the completeness of our method. To check the optimality of each algorithm, we measure a cost error that quantifies the discrepancy between the optimal cost found by CBS and the total cost obtained by other algorithms, i.e., $(\text{solution cost} - \text{optimal cost})/\text{optimal cost}$, where the cost represents total paths length. Hence, lower cost error indicates better performance. These metrics were averaged across test scenarios.

Fig. 6 illustrates the effectiveness of using the conflict map. (b) and (c) show the initial A* result and results of CBS without the conflict map. In contrast, (e), and (f) show the initial A* result and results of CBS when using the conflict map. In (d), the red area indicates the conflict map of the red agent, demonstrating that the red agent has selected a bypass of the same length in (e)

Table I shows the results obtained from the 32×32 scenarios. A comparison with CBS reveals that our approach yields improvements in constraint tree size and computation time. Our method reduces approximately 40% in computation time relative to CBS. While ICBS showcases a reduction in tree size, it is observed to exhibit increased computation time. An apparent reduction in computation time compared to CBS is evident in learning-based methods. Our method outperforms PRIMAL in speed and performs slightly slower than DHC. DHC demonstrated approximately a 30% faster computation time compared to our method. However, it is worth noting that PRIMAL and DHC fail to achieve a 100% success rate, with PRIMAL deviating by 18.7% and DHC by 8.1% from the optimal path cost.

In a 64×64 scenario, our method achieves a computation time reduction of approximately 40% compared to CBS. The results presented in Table II show similar trends to those observed in the 32x32 scenario. Once again, our approach demonstrates reductions in both tree size and computation time relative to CBS, while ICBS achieves a decrease in tree size but an increase in computation time. Compared to learning-based methods, our approach is slightly slower than the two methods for the 64×64 scenario. DHC exhibited approximately 60% faster computation time compared to our method. The success rate of learning-based methods was above 97%, and the cost error for PRIMAL was 19.7%, while for DHC, it was 8.9%. Cost error slightly increases compared to the results obtained for the 32×32 scenario.

Metrics	CBS	ICBS	DHC	PRIMAL	Proposed
n_{Gen}	557	434	N/A	N/A	366
n_{Exp}	279	30	N/A	N/A	183
CPU Time (s)	4.4	19	**1.8**	3.7	2.6
Success Rate (%)	100	100	99.9	91.3	100
Cost Error (%)	0	0	8.1	18.7	0

TABLE I: Performance comparison in 32×32 Scenarios. The average optimal cost is 434.

Metrics	CBS	ICBS	DHC	PRIMAL	Proposed
n_{Gen}	1074	173	N/A	N/A	444
n_{Exp}	537	90	N/A	N/A	223
CPU Time (s)	12.7	19.3	**3.1**	6.1	7.6
Success Rate (%)	100	100	99.4	97.4	100
Cost Error (%)	0	0	8.9	19.7	0

TABLE II: Performance comparison in 64×64 Scenarios. The average optimal cost is 869.

Metrics	CBS	ICBS	DHC	PRIMAL	Proposed
n_{Gen}	409	97	N/A	N/A	81
n_{Exp}	205	51	N/A	N/A	41
CPU Time (s)	35.4	77.2	**5.5**	19.6	6.3
Success Rate (%)	100	100	94.2	99.5	100
Cost Error (%)	0	0	5.4	15.3	0

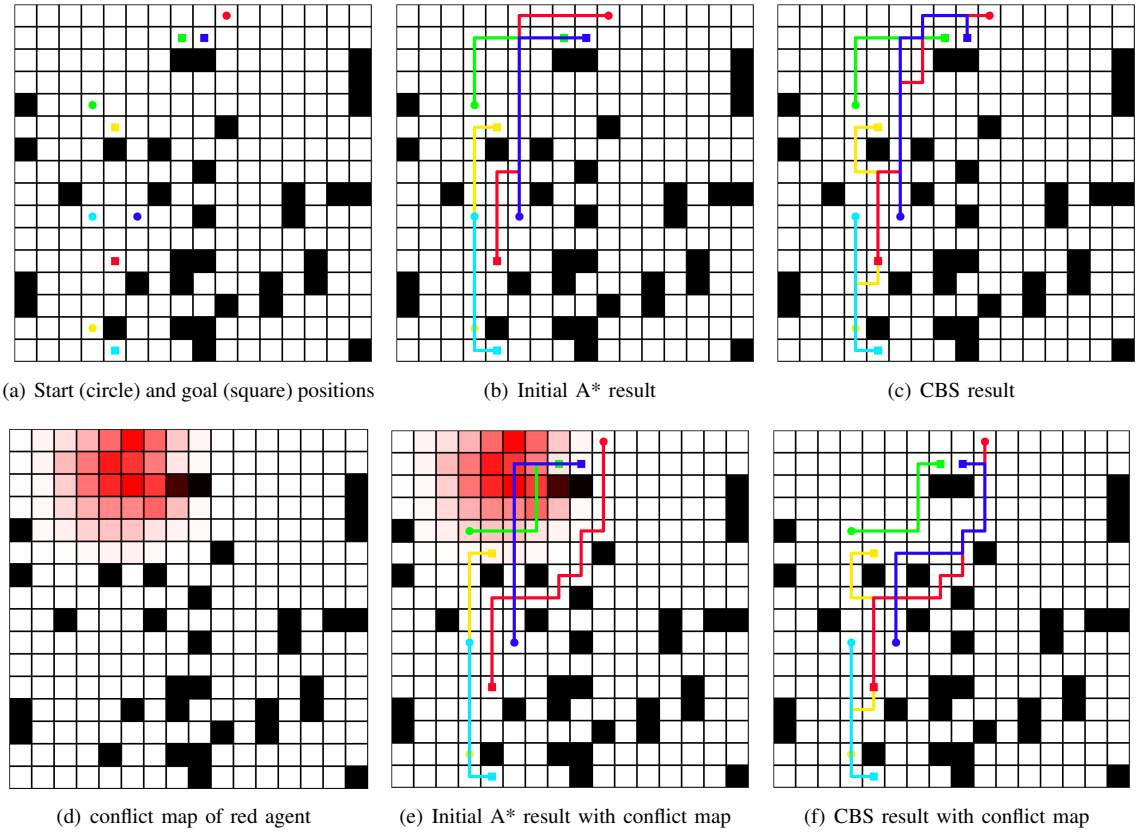
TABLE III: Performance comparison in 128×128 Scenarios. The average optimal cost is 1714.

Metrics	CBS	ICBS	DHC	PRIMAL	Proposed
n_{Gen}	150	27	N/A	N/A	58
n_{Exp}	75	23	N/A	N/A	29
CPU Time (s)	48.1	159.5	**12.9**	189.6	29.9
Success Rate (%)	100	100	44.7	1.3	100
Cost Error (%)	0	0	1.1	50.4	0

TABLE IV: Performance comparison in 256×256 Scenarios. The average optimal cost is 7859.

Table III presents the results obtained from the 128×128 scenario. The comparisons with search-based methods exhibit similar trends to previous scenarios. In the 128×128 scenario, the proposed method achieves an approximate 80% reduction in computation time, which can also be attributed to the scaling-up effect. The computation time of DHC is approximately 13% faster than our proposed method. DHC exhibited a lower success rate, while PRIMAL achieved a high success rate of 99.5%. Furthermore, the cost error decreased compared to the previous scenario, with PRIMAL at 15.3% and DHC at 5.4%.

Table IV is the results in the 256×256 scenario. The results of search-based methods exhibit a similar pattern to previous scenarios. Notably, the node count is lower than in the 128×128 scenario, influenced by empty spaces in the Boston scenario. Despite the reduced node count, computation time has overall increased, attributed to the larger map size and a more significant number of agents, placing a higher load on centralized methods. Our proposed approach achieves around a 38% reduction in computation time compared to CBS. Learning-based methods exhibit



(a) Start (circle) and goal (square) positions (b) Initial A* result (c) CBS result

(d) conflict map of red agent (e) Initial A* result with conflict map (f) CBS result with conflict map

Fig. 6: Illustration of comparing paths with and without utilizing the conflict map in CBS.

significantly reduced performance in the 256×256 scenario. When aggregating the results from successful scenarios, DHC is approximately 55% faster than our proposed method. DHC and PRIMAL often encounter difficulties in scenarios they fail to solve, repeatedly traversing long obstacles. This behavior likely results from the observation size of agents being insufficient to cover the extensive obstacles in the Boston scenario. The higher success rate of DHC was achieved by using the heuristic map as a global guidance information different from the goal vector of PRIMAL and using a communication module. The cost error for successful scenarios was 50.4% for PRIMAL and 1.1% for DHC.

Through experiments, we conclude that our proposed method can reduce computation time compared to CBS. The extent of this reduction was around 40% in most scenarios, excluding the larger 128×128 scenario where the reduction reached 80%. This reduction in computation time was achieved without compromising the optimality and completeness of CBS. Additionally, we conducted comparisons with state-of-the-art learning-based methods, DHC and PRIMAL, as well as distributed methods. Our proposed approach achieved faster computation times than PRIMAL in all scenarios except 64×64. Compared to DHC, our method demonstrated approximately 45% slower in most scenarios. Nevertheless, when considering the success rates and cost errors, our approach reduces the computation time without compromising optimality and completeness.

In fact, directly comparing learning-based methods with search-based methods can be unfair since learning-based methods usually train controllers instead of planners, which can be directly used to control multi-agents. From this perspective, search-based methods have disadvantages in that all agents must stop until solution paths are found. Nevertheless, through experimental results of computation time and success rate, we have observed that as the map size increases, learning-based methods tend to fail in finding feasible and optimal paths. Based on these observations, we believe that accelerating search-based methods has a clear benefit over learning-based methods.

VI. CONCLUSION

We introduced a novel approach to reduce computation time for solving traditional MAPF problems using a deep learning. Our proposed method involves predicting conflict-prone areas using a neural network to generate the conflict map which is integrated into the A* algorithm. By incorporating the conflict map, CBS is encouraged to consider alternative paths that avoid areas where conflicts are anticipated, ultimately leading to reduced computation time. We conducted comparative experiments with various search-based and learning-based MAPF methods. While our approach often exhibited slower computation times compared to the state-of-the-art learning-based method, DHC, our experiments confirmed that our method does not compromise optimality and completeness and can reduce computation time for search-based methods without sacrificing these properties.

REFERENCES

[1] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics Auton. Syst.*, vol. 41, no. 2-3, pp. 89–99, 2002.

[2] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, pp. 9–20, 2008.

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

[4] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[5] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, M. Fox and D. Poole, Eds. AAAI Press, 2010, pp. 173–178.

[6] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*. IEEE, 2011, pp. 3260–3267.

[7] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 195, pp. 470–495, 2013.

[8] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *PRICAI 2012: Trends in Artificial Intelligence - 12th Pacific Rim International Conference on Artificial Intelligence, Kuching, Malaysia, September 3-7, 2012. Proceedings*, ser. Lecture Notes in Computer Science, P. Anthony, M. Ishizuka, and D. Lukose, Eds., vol. 7458. Springer, 2012, pp. 564–576.

[9] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, M. desJardins and M. L. Littman, Eds. AAAI Press, 2013.

[10] D. Sigurdson, V. Bulitko, S. Koenig, C. Hernández, and W. Yeoh, "Automatic algorithm selection in multi-agent pathfinding," *CoRR*, vol. abs/1906.03992, 2019.

[11] 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, 2015.

[12] 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 Robotics Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, 2019.

[13] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 8699–8705.

[14] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and S. E. Shimony, "ICBS: improved conflict-based search algorithm for multi-agent pathfinding," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M. J. Wooldridge, Eds. AAAI Press, 2015, pp. 740–746.

[15] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, ser. Frontiers in Artificial Intelligence and Applications, T. Schaub, G. Friedrich, and B. O'Sullivan, Eds., vol. 263. IOS Press, 2014, pp. 961–962.

[16] D. Silver, "Cooperative pathfinding," in *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, R. M. Young and J. E. Laird, Eds. AAAI Press, 2005, pp. 117–122.

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

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

[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[20] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Barták, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, P. Surynek and W. Yeoh, Eds. AAAI Press, 2019, pp. 151–159.