한국정밀공학회지 제 43 권 제 4 호 pp. 317-324 April 2026 / 317
J. Korean Soc. Precis. Eng., Vol. 43, No. 4, pp. 317-324 http://doi.org/10.7736/JKSPE.025.045
ISSN 1225-9071 (Print) / 2287-8769 (Online)

연속 시간을 이용한 공간 활용을 통해 개선된 충돌 기반 탐색에 관한 연구

A Study on Improving Conflict Based Search with Continuous Time Using Space Utilization

임성택[1], 유서현[1], 강효재[1], 정찬희[2], 한대희[1], 강민성[3,#]
SeongTaek Im[1], SeoHyun Yoo[1], HyoJae Kang[1], ChanHui Jung[2], DaeHee Han[1], and Min-Sung Kang[3,#]

1 한양대학교 융합로봇시스템학과 (Department of Interdisciplinary Robot Engineering Systems, Hanyang University)
2 한양대학교 로봇공학과 (Department of Robot Engineering, Hanyang University)
3 한양대학교 스마트융합공학부 (School of Smart Convergence Engineering, Hanyang University)
Corresponding Author / E-mail: wowmecha@hanyang.ac.kr, Tel: +82-31-400-5961
ORCID: 0000-0002-8459-5843

KEYWORDS: Multi-agent path finding (다중 에이전트 경로 탐색), Collision based search (충돌 기반 탐색), Path planning (경로 탐색), Robot navigation (로봇 네비게이션)

Multi-Agent Path Finding (MAPF) is an algorithm designed to identify collision-free paths for multiple agents, commonly used in fields like robotics and drone navigation. Conflict-Based Search with Continuous Time (CCBS) is particularly beneficial for real-world applications due to its capability to find paths in continuous time; however, it often experiences lengthy computation times. Although techniques such as prioritizing conflicts (PC), disjoint splitting (DS), and high-level heuristics have been implemented to reduce these times, challenges remain. To address these issues, this paper introduces methods to improve space utilization by calculating agent congestion. By optimizing space usage, we can identify paths that avoid potential collisions, even when those paths share the same cost. We propose enhancements to high-level heuristics, conflict prioritization, and low-level heuristics, as well as a method for calculating congestion in continuous time. These improvements lead to a reduction in agent collisions and a decrease in high-level expansions, resulting in a 30% increase in computational success rates compared to the existing CCBS. Incorporating space utilization into the search process significantly enhances MAPF performance.

Manuscript received: March 25, 2025 / Revised: November 19, 2025 / Accepted: November 24, 2025
This paper was presented at KSPE Autumn Conference in 2024

1. Introduction

The Multi-Agent Path Finding (MAPF) problem is accompanied by multiple agents, each having a designated start and goal location. The challenge is to find paths for each agent without collisions. MAPF can be applied in scenarios like warehouse management where paths for many agents need to be determined [1,2]. One of the optimal solutions for the MAPF problem is minimizing the cost (Sum of Cost) [3]. However, this problem is riddled with challenges of complexity and nonlinearity, making it NP-hard [4,5].

To address this issue, numerous AI researchers have attempted solutions through various prior works. However, most al- gorithms solve the problem by assuming 1) time discretization and 2) the duration of each action is one time step [6-8]. Such assumptions can become limitations when applied to robots in real-world settings. To operate actual robots, one must consider kinematic constraints such as rotation time and robot localization. Therefore, to overcome these challenges, several methods, like MAPF-Post, have been proposed

to address and surmount these limitations [9,10].

CCBS was developed to address the MAPF problem in continuous time and make MAPF suitable for real-world environments [11]. By integrating Safe Interval Path Planning (SIPP) into the low-level search of CBS (Conflict-Based Search), the algorithm helps MAPF operate in continuous time [12,13]. However, one major drawback of CCBS is that detecting collisions in continuous time requires a considerable amount of computational resources. To address this weakness, several algorithms have been proposed to minimize conflicts among agents to reduce computation time.

For example, Conflict-Based Search with Real-Time Conflict Resolution (CBS-RTC) introduced new types of conflicts to address specific MAPF conditions, such as corridor and target symmetry [14]. Furthermore, techniques like Disjoint Splitting (DS), Prioritizing Conflicts (PC), and enhanced heuristics are also applied to this algorithm to minimize the search time [15-17]. These methods enhanced the efficiency of CCBS by minimizing conflicts at higher-level nodes [18].

Han *et al*. pointed out, as illustrated in Fig. 1, that reuse of the same areas during computing individual agents' paths frequently occurs in MAPF algorithms, which can cause inefficiencies in the algorithm [19]. To optimize space utilization, a heuristic function that estimates the redundancy of vertices and edges in the agent's paths found by agents was proposed. This approach has been adopted by other algorithms like ECBS and DDM, enabling significant improvements in the computational performance while maintaining a similar level of optimality [20,21].

In addition, Chen *et al*. [22] introduced an algorithm that utilizes traffic flow to create guide paths in priority-based algorithms like PIBT, LaCAM* [23,24], and MAPF-LNS to alleviate congestion [25-27]. These methods are more efficient even in jammed circumstances and show improvements in overall system performance. Therefore, one effective way to increase the success rate in MAPF is maximizing space usage. This paper proposes a method that calculates congestion and applies to the algorithm to enhance the computational performance and success rate of Improving CCBS. The goal is to minimize potential conflicts between agents during the search process by optimizing space utilization, ultimately reducing the branching of high-level nodes. To achieve this, we introduced a heuristic function for high-level nodes, improved the method for prioritizing conflicts, and proposed a heuristic function for low-level nodes. These approaches have led to an improvement in space utilization during the computation process.

In this study, experiments are conducted on diverse grid maps (empty-16-16, warehouse-10-20-10-2-2, den520d). The results showed that the method significantly reduced the branching of high-level nodes at least 10 times and up to 100 times compared to Improving CCBS by minimizing conflicts between agents. This reduction in the number of branches resulted in an average increase in success rate of approximately 30% across different grid maps. Moreover, the increase in the flow time can be negligible relative to prior algorithms, indicating that the proposed method is both efficient and practical.

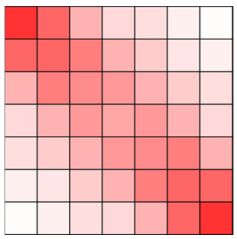

Fig. 1 Overuse of free space

2. Theoretical Background

2.1 Conflict Based Search with Continuous Time (CCBS)

CCBS is an algorithm designed to solve MAPF under continuous time, while considering kinematic characteristics such as rotation and acceleration of robots. The process of this algorithm is divided into a high-level stage and a low-level stage. The high-level stage detects conflicts between agents and generates constraints to resolve them, inheriting these from parent nodes to child nodes. Sequentially, the child nodes solve these conflicts through the low-level algorithm, Safe-Interval Path Planning (SIPP).

SIPP optimizes a single agent’s path by categorizing between Safe Intervals and Unsafe Intervals based on the provided constraints. The agent is allowed to move or stop only within Safe Intervals, ensuring collision-free path planning. Although CCBS effectively finds optimal paths, its conflict detection process can be computationally expensive.

To enhance CCBS, Andreychuk *et al*. incorporated successful improvements from CBS. They introduced Disjoint Splitting (DS) to efficiently distinguish the search space using both Positive and Negative Constraints, guaranteeing that child nodes explore mutually exclusive solutions. This strategy noticeably boosts the search efficiency.

Furthermore, they introduced Prioritizing Conflicts (PC) to efficiently identify and resolve the most critical conflicts, reducing the number of expanded nodes and optimizing search performance. Finally, heuristic techniques were implemented in the high-level

search to improve the computation speed by estimating potential cost increases and prioritizing nodes with the highest impact on conflict resolution, thereby reducing unnecessary node expansions and enabling faster optimal pathfinding.

2.2 Space Utilization Optimization (SU-I)

SU-I is a heuristic designed to optimize space utilization in Multi-Robot Path Planning (MRPP) and Life-Long Multi-Robot Path Planning (LMPP). Its primary goal is to navigate robots while using space evenly during path planning, reducing conflicts and enabling more efficient discovery of optimal routes.

SU-I considers vertices, edges, and temporal information of the graph, directing paths stay away from congested areas. It tracks how much each part of the graph is used by other robots, helping them avoid heavily used routes and select paths with less traffic.

SU-I can be integrated into the A^* algorithm in two main ways: by incorporating it into the estimated cost-to-go to reduce potential conflicts, or into the cost-to-come to minimize actual conflicts along the path. These methods enable SU-I to achieve both path optimality and conflict avoidance simultaneously.

Additionally, SU-I anticipates interactions between robots' paths and selects routes that minimize these events, making it particularly effective in complex environments. The heuristic also improves path selection accuracy through multiple planning iterations, updating graph space utilization and continuously reducing path conflicts, thereby enhancing the overall efficiency of the process.

3. Methodology

As mentioned by Han *et al*. CCBS has the drawback of insufficient space utilization during the search process for agents. For example, as illustrated in Fig. 2(a), if the optimal single path is searched without considering the paths of other agents, collisions may occur. To resolve these collisions, the conflicts are set as constraints, leading to branching in the high-level nodes, and the SIPP is used for the single search of the robot based on the given constraints. However, if space utilization is insufficient, frequent branching of high-level nodes occurs due to collisions, leading to increased computational load. On the other hand, when search is conducted with space utilization in mind, it is possible to find a collision-free path with the same cost, as shown in Fig. 2(b).

The importance of space utilization and its impacts on overall algorithm performance have been emphasized in various studies. In this context, we propose three key

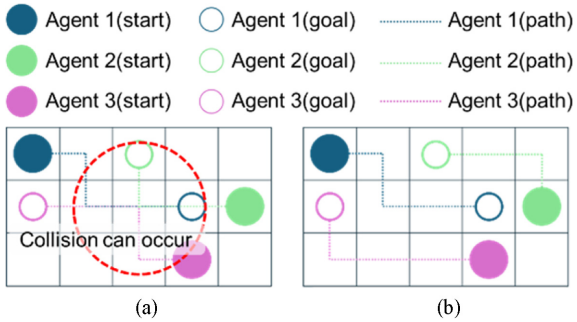

Fig. 2 Space utilization example. (a) without spatial utilization, (b) with spatial utilization

improvements to address this issue in CCBS: improving the High-Level heuristic function, refining the method for determining cardinality, and enhancing the Low-Level heuristic function. The purpose of these keys is to maximize space utilization, resulting in increasing the success rate of search and reducing the computational load.

Before presenting Algorithms, we first introduce several terms and notation used in this section. An agent's path is expressed as a sequence of nodes $p[i]$, where each node corresponds to the i-th position the agent reaches during its movement. For each node n_1, we denote its arrival time as $n_1.g$, and its spaticl location on the grid map as $(n_1.x, n_1.y)$. Using these node states, the algorithm constructs a congestion array T, which records how many agents occupy each grid cell over time and is used to evaluate congestion-based heuristic values. The notation v_{pi}, refers to the vertex currently occupied by agent i, and p denotes the path taken by an agent. We use n to denote the total number of agents in the scenario, and d denotes the Euclidean distance between the current node and the goal node, which is used as part of the heuristic evaluation. Finally, h denotes the heuristic value computed during the search process.

3.1 Congestion Calculation for Space Utilization

This paper proposes a method to calculate a congestion array based on the movement paths of agents. Various methods have been proposed in previous studies to optimize space utilization, and in this study, we adopt an approach that calculates a congestion array based on the movement paths of each agent. For example, when an agent moves from Node N_a to Node N_b, the congestion at N_a is recorded over time, and this congestion is reflected in the continuous flow of time. To do this, the time is divided into intervals, and the time of arrival at N_a is recorded in each interval, incrementing the congestion value of N_a by 1. This process is uniformly applied to the paths of all agents, ultimately calculating the congestion array.

Algorithm 1: Save congestion array

```
1   Congestion ← {}
2   for all p in Path do
3   for i =1 to ; length(p) − 1 do
4     n1 = p[i], n2 = p[i + 1]
5     t1 = n1.g/timestep, t2 = n2.g/timestep
6     Congestion[n1.x][n2.y][t1] += 1
7     Calculate intermediated node between n1 and n2
8   for each intermediate node between n1 and n2 do
9     Congestion[x][y][t] += 1
10   end for
11   Congestion[n2.x][n2.y][t2] +=1
12  end for
13  end for
```

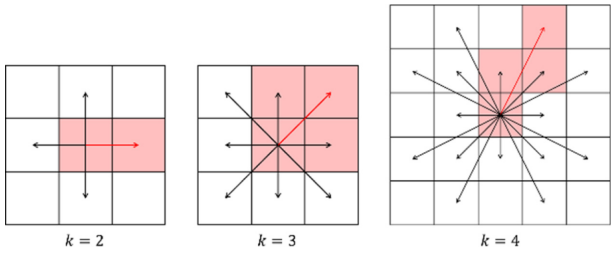

Fig. 3 Congestion increase according to the illustration of the 2^k neighborhood for $k = 2$, 3 and 4

To effectively utilize the congestion array, the size and speed of each agent, as well as changes in the path over time, must be considered. Specifically, when an agent moves from Node N_a to Node N_b, the nodes located at intermediate points should also be included in the congestion array to accurately model the space that the agent actually occupies.

The robot's movement path is determined by a 2^k neighborhood structure, as shown in Fig. 3. When the robot moves along the red arrows, all the red nodes along the path are included in the congestion calculation. This plays a crucial role in predicting all potential collision points on the path in advance and designing the path to avoid them.

Previous research applied methods to balance the search by setting the sum of vertex and edge congestion weights to 1. However, this study improves this methodology by adopting an approach that solely considers vertex congestion when finding the optimal path. This enables a more detailed analysis of the space that the agent actually occupies on the path, enhancing the accuracy and efficiency of the search. This congestion-based approach contributes not only to optimizing part of the search process but also to enhancing the efficiency of the overall search algorithm.

Algorithm 2: Get high-level congestion cost

```
1   cost ← 0
2   for all p in Path do
3   for i = 1 to length(p) − 1 do
4    n1 = p[i], n2 = p[i + 1]
5     t1 = n1.g/timestep, t2 = n2.g/timestep
6   cost ← cost + Congestion[n1][n2][t1]
7     Calculate intermediate nodes between n1 and n2
8     for each intermediate node between n1 and n2 do
9       cost ← cost + Congestion[x][y][t]
10    end for
11  cost ← cost + Congestion Congestion
                  [n2.x][n2.y][t2]
12  end for
13  end for
14  return cost/length(Path)
```

3.2 Congestion-based High-level Heuristic Function for Space Utilization

Andreychuk *et al.* applied a heuristic to CCBS using a Linear Programming Problem (LPP) to estimate the minimum cost increase necessary to resolve conflicts and to identify the increase in solution cost during conflict resolution, thereby minimizing computation time. In this study, we expand this approach by applying a new heuristic function that incorporates the calculated congestion array. This congestion array stores values that represent the density of agents passing through specific nodes during a given time interval. Summing the congestion of each node and dividing by the number of agents calculates the average congestion to incorporate into the heuristic function. This guides the search process to avoid highly congested areas, thereby minimizing potential collisions and improving space utilization. Therefore, this approach considers all possibilities from the initial stage of the search to compute more optimized paths. For instance, by guiding the initial path configuration to avoid highly congested areas, potential collisions in the intermediate stages of the search can be prevented. This strategy significantly increases the success rate of search, especially when multiple agents move simultaneously in complex environments, through the use of space-utilized heuristic functions.

3.3 Prioritizing Conflicts for Space Utilization

The Prioritizing Conflicts is a method for determining which conflict should be resolved first when conflicts happen between agents. This method classifies conflicts into three types: Cardinal Conflicts, Semi-Cardinal Conflicts, and Non-Cardinal Conflicts,

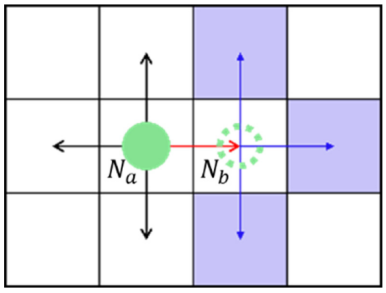

Fig. 4 Low level search heuristic calculation example

based on whether resolving the conflict causes an increase in the cost of the child nodes relative to the parent node. Cardinal Conflicts take place when the cost of the child nodes is higher than that of the parent node, Semi-Cardinal Conflicts occur when the cost of one child node's increases while the other remains the same, and Non-Cardinal Conflicts occur when the cost does not increase. PC primarily prioritizes conflicts based on path costs, addressing the highest-priority conflicts first to enhance the overall efficiency of search. In this approach, the path cost is determined by summing the path costs of all agents and then calculating the difference from the previous path costs to determine the priority of each conflict.

However, the traditional PC approach has limitations as it only considers costs without sufficiently reflecting space utilization. In this study, we propose an enhanced conflict prioritization method that incorporates congestion, considering both cost and space utilization simultaneously. To achieve this, we calculate the congestion-based increase in value after determining each agent's path. If all child nodes have higher congestion and path costs compared to their parent nodes, the conflict is defined as a Cardinal Conflict. If only one child node shows an increase in both congestion and path cost, it is classified as a Semi-Cardinal Conflict. If none of the child nodes meet the conditions for congestion and path cost increase, it is classified as a Non-Cardinal Conflict.

This method comprehensively considers both the path cost and congestion increase of the child nodes to determine the optimal conflict prioritization. An increase in congestion directly impacts space utilization, making conflict resolution that accounts for this factor more efficient during the search.

Additionally, the cost increase is defined as the sum of the changes in path cost and congestion cost. By resolving not only conflicts with the greatest impact on the total solution cost but also those with a huge impact on congestion, the number of expanded nodes at the High-level Node can be reduced. This approach plays a critical role in solving multi agent search problems in dense environments, resulting in improved computational efficiency and enhanced solution quality.

Algorithm 3: Get low-level congestion cost

1 $moves \leftarrow \{\}$
2 $cost \leftarrow 0$
3 **for** *each* m in $moves$ **do**
4 if $m.id == current\ Node.id$ **then**
5 continue
6 end if
7 $distance \leftarrow \sqrt{(m.i - goal.i)^2 + (m.j - goal.j)^2}$
8 $cost \leftarrow cost + \left(\frac{c\ array\left[m \cdot \frac{g}{timestep} \right][m.i][m.j]}{distance} \right)$
9 end for
10 $h \leftarrow h + \left(\frac{cost}{moves.size()} \right)$

3.4 Congestion-base Low-level Search Heuristic Function for Space Utilization

This paper proposes a method that incorporates congestion into the heuristic function of the Low-Level Search to preemptively identify paths for a single agent that will not cause conflicts with the paths of other agents. This method focuses on identifying paths that can avoid potential conflicts with other agents, even when the costs are identical, by considering congestion. To accomplish this, the congestion at each potential node is calculated and reflected in the heuristic value during the search process for a single agent. Specifically, as the agent evaluates potential paths from one node to the next, the congestion of the surrounding nodes is measured and integrated into the heuristic. For example, as shown in Fig. 4, if an agent is evaluating moving from Node N_a to Node N_b, the congestion around N_b is assessed by inspecting the nodes in the blue region. This process predicts the congestion in the future paths and determines the feasibility of moving to N_b.

Congestion calculation is performed using a Congestion Array. The congestion around N_b is computed by considering the congestion levels of nodes accessible from N_b, based on the Congestion Array, and then dividing this value by the distance from N_b to the goal node. The heuristic value is determined by summing the congestion of all nodes and then dividing by the number of accessible nodes. This approach enables the anticipation of future congestion during the search process, allowing the agent to avoid moving to nodes with high surrounding congestion. This is particularly effective in preventing conflicts when multiple agents navigate in complex environments simultaneously. Therefore, incorporating congestion into the Low-level Search improves space utilization and increases search success rates

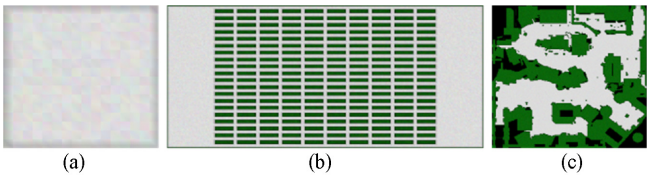

Fig. 5 Example of grid map for evaluation. (a) empty-16-16, (b) warehouse-10-20-10-2-2, (c) den520d

compared to traditional search algorithms, while minimizing conflicts between agents.

4. Experiments

4.1 Evaluation Setup

To evaluate the impact of the proposed heuristic modifications on CCBS-DS-PC-H, we conducted extensive experiments ac- ross various MAPF scenarios. We compared the conges- tion-based space utilization heuristic with the H2 heuristic proposed in previous research. The experiments were conducted on three grid map environments shown in Fig. 5: empty-16-16, warehouse-10-20-10-2-2, and den520d.

- **empty-16-16**: A grid map with wide-open spaces to evaluate performance in low density, unobstructed environments
- **warehouse-10-20-10-2-2**: A structured grid resembling a warehouse with narrow aisles, representing high-density environments
- **den520d**: A complex, obstacle-rich map to assess the algorithm's robustness in dynmic and cluttered scenarios

In each scenario, the agents were modeled as disk-shaped entities with a radius of $\sqrt{2}/4$. The time limit for each computation was set to 1 minute. The experiments were conducted on a computer with a Ryzen 5800x CPU and 32GB RAM. The neighborhood structure was defined by a 2^k neighborhood, with k set to 3 and 5 for comparative analysis. The primary metrics used for comparison were the number of CT (Constraint Tree) node branches and the success rate of the computations.

4.2 Evaluation Results and Analysis

The experimental results illustrate the superior performance of CCBS with the proposed congestion-based space utilization heuristic. As shown in Fig. 6, the proposed heuristic maps, demonstrating the consistent superiority of the congestion-based approach.

This performance improvement is further validated by the results shown in Fig. 7 where the number of CT node branches are compared. The results refer to the congestion-based heuristic that

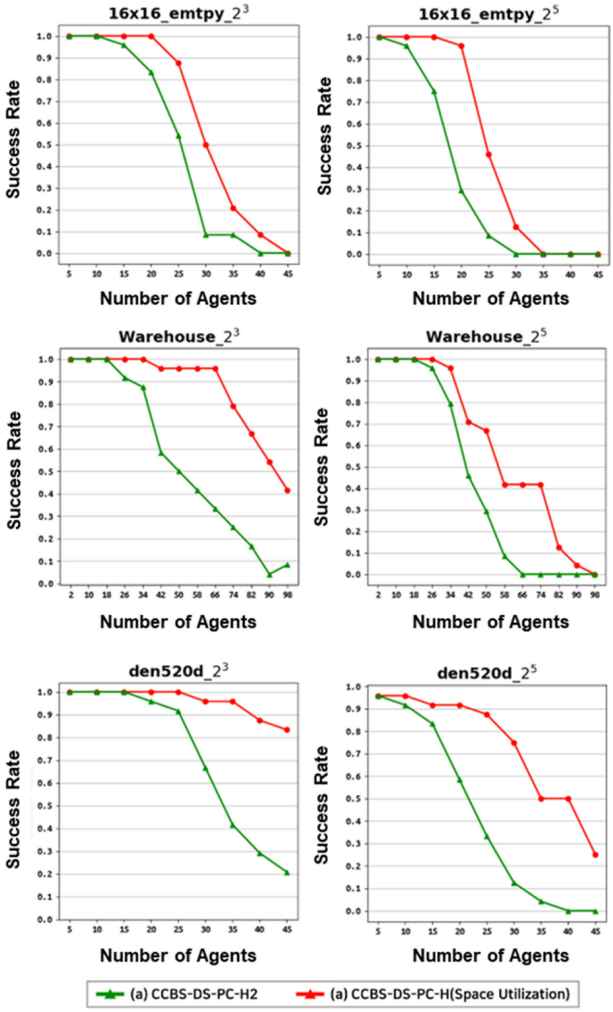

Fig. 6 Success rates for CCBS and its modifications on different 2^k-connected grids

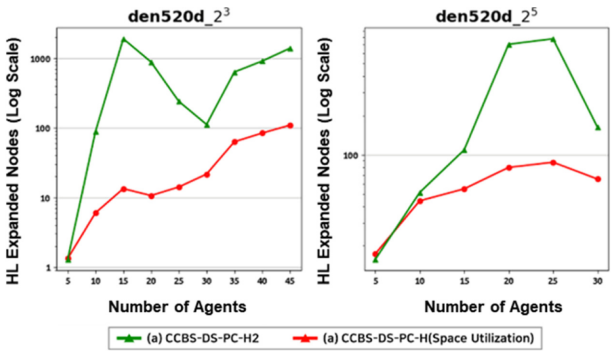

Fig. 7 Expanded CT nodes for solved instance of CCBS

reduces the number of branches by up to 100 times in certain cases, with a minimum reduction of at least 10 branches. This reduction in branching can be attributed to the increased space utilization, which lowers the likelihood of collisions with other agents. These conclusions suggest that broad space utilization during pathfinding can significantly reduce the risk of future collisions.

The results of this experiment demonstrate the impact of integrating congestion-based heuristics into CCBS. By prioritizing space utilization, the proposed approach improves computational efficiency, increasing the success rate of pathfinding in complex environments. These findings highlight the necessity of space utilization in future research of multi-agent pathfinding algorithms, especially in environments with high agent density or numerous obstacles.

5. Conclusion

In this study, we introduced an approach to optimize space utilization in the enhanced CCBS algorithm to increase the success rate of MAPF. By introducing congestion-based heuristics for both high-level and low-level nodes and refining the conflict prioritization method. We effectively reduced the likelihood of collisions between agents, including a significant reduction in the branching of high-level nodes and improved the computational efficiency of the algorithm. Experiments conducted on various grid maps, including empty-16-16, warehouse-10-20-10-2-2, and den520d, demonstrated that the proposed method significantly outperformed the existing Improving CCBS. Specifically, the proposed method reduced high-level node branching by at least 10 times and up to 100 times, while increasing the algorithm's success rate by over 30% across different scenarios. Additionally, the impact on flow time relative to existing algorithms was minimal, confirming the practicality and effectiveness of the proposed approach. These results emphasize the importance of space utilization in MAPF algorithms, particularly in environments with high agent density or complex obstacles. Future research could explore the development of space utilization-based constraints, guide path, and the integration of congestion-based heuristics into other MAPF algorithms, potentially leading to even greater performance improvements and higher success rates in real-world applications.

REFERENCES

1. Wurman, P. R., D'Andrea, R., Mountz, M., (2008), Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Magazine, 29(1), 9-9.
2. Morris, R., Pasareanu, C. S., Luckow, K. S., Malik, W., Ma, H., Kumar, T. S., Koenig, S., (2016), Planning, scheduling and monitoring for airport surface operations, Proceedings of the Workshops of the Thirtieth AAAI Conference on Artificial Intelligence Planning for Hybrid Systems.
3. Dresner, K., Stone, P., (2008), A multiagent approach to autonomous intersection management, Journal of Artificial Intelligence Research, 31, 591-656.
4. Yu, J., LaValle, S., (2013), Structure and intractability of optimal multi-robot path planning on graphs, Proceedings of the AAAI Conference on Artificial Intelligence, 27(1), 1443-1449.
5. Badawy, M., Khalifa, H., Arafat, H., (2019), New approach to enhancing the performance of cloud-based vision system of mobile robots, Computers & Electrical Engineering, 74, 1-21.
6. Sharon, G., Stern, R., Felner, A., Sturtevant, N. R., (2015), Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence, 219, 40-66.
7. Sharon, G., Stern, R., Goldenberg, M., Felner, A., (2013), The increasing cost tree search for optimal multi-agent pathfinding, Artificial Intelligence, 195, 470-495.
8. Goldenberg, M., Felner, A., Stern, R., Sharon, G., Sturtevant, N., Holte, R. C., Schaeffer, J., (2014), Enhanced partial expansion A, Journal of Artificial Intelligence Research, 50, 141-187.
9. Hönig, W., Kumar, T., Cohen, L., Ma, H., Xu, H., Ayanian, N., Koenig, S., (2016), Multi-agent path finding with kinematic constraints, Proceedings of the International Conference on Automated Planning and Scheduling, 26, 477-485.
10. Ma, H., Hönig, W., Kumar, T. S., Ayanian, N., Koenig, S., (2019), Lifelong path planning with kinematic constraints for multi-agent pickup and delivery, Proceedings of the AAAI Conference on Artificial Intelligence, 7651-7658.
11. Andreychuk, A., Yakovlev, K., Surynek, P., Atzmon, D., Stern, R., (2022), Multi-agent pathfinding with continuous time, Artificial Intelligence, 305, 103662.
12. Phillips, M., Likhachev, M., (2011), Sipp: Safe interval path planning for dynamic environments, Proceedings of the IEEE International Conference on Robotics and Automation, 5628-5635.
13. Sharon, G., Stern, R., Felner, A., Sturtevant, N. R., (2015), Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence, 219, 40-66.
14. Li, J., Harabor, D., Stuckey, P. J., Ma, H., Gange, G., Koenig, S., (2021), Pairwise symmetry reasoning for multi-agent path finding search, Artificial Intelligence, 301, 103574.
15. Li, J., Harabor, D., Stuckey, P. J., Felner, A., Ma, H., Koenig, S., (2019), Disjoint splitting for multi-agent path finding with conflict-based search, Proceedings of the International Conference on Automated Planning and Scheduling, 29, 279-283.
16. Boyarski, E., Felner, A., Stern, R., Sharon, G., Betzalel, O., Tolpin, D., Shimony, E., (2015), Icbs: The improved conflict-based search algorithm for multi-agent pathfinding, Proceedings of the International Symposium on Combinatorial Search, 6(1), 223-225.

17. Li, J., Felner, A., Boyarski, E., Ma, H., Koenig, S., (2019), Improved heuristics for multi-agent path finding with conflict-based search, Proceedings of the International Joint Conferences on Artificial Intelligence, 442-449.

18. Andreychuk, A., Yakovlev, K., Boyarski, E., Stern, R., (2021), Improving continuous-time conflict based search, Proceedings of the AAAI Conference on Artificial Intelligence, 11220-11227.

19. Han, S. D., Yu, J., (2022), Optimizing space utilization for more effective multi-robot path planning, Proceedings of the International Conference on Robotics and Automation, 10709-10715.

20. Han, S. D., Yu, J., (2020), Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics, IEEE Robotics and Automation Letters, 5(2), 1350-1357.

21. Barer, M., Sharon, G., Stern, R., Felner, A., (2014), Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem, Proceedings of the International Symposium on Combinatorial Search, 19-27.

22. Chen, Z., Harabor, D., Li, J., Stuckey, P. J., (2024), Traffic flow optimisation for lifelong multi-agent path finding, Proceedings of the AAAI Conference on Artificial Intelligence, 38(18), 20674-20682.

23. Okumura, K., Machida, M., Défago, X., Tamura, Y., (2022), Priority inheritance with backtracking for iterative multi-agent path finding, Artificial Intelligence, 310, 103752.

24. Okumura, K., (2023), Lacam: Search-based algorithm for quick multi-agent pathfinding, Proceedings of the AAAI Conference on Artificial Intelligence, 37(10), 11655-11662.

25. Li, J., Chen, Z., Harabor, D., Stuckey, P. J., Koenig, S., (2021), Anytime multi-agent path finding via large neighborhood search, Proceedings of the International Joint Conference on Artificial Intelligence, 4127-4135.

26. Li, J., Chen, Z., Harabor, D., Stuckey, P. J., Koenig, S., (2022), Mapf-lns2: Fast repairing for multi-agent path finding via large neighborhood search, Proceedings of the AAAI Conference on Artificial Intelligence, 36(9), 10256-10265.

27. Li, J., Chen, Z., Zheng, Y., Chan, S.-H., Harabor, D., Stuckey, P. J., Ma, H., Koenig, S., (2021), Scalable rail planning and replanning: Winning the 2020 flatland challenge, Proceedings of the International Conference on Automated Planning and Scheduling, 31, 477-485.

SeongTaek Im
M.Sc. candidate in the Department of Interdisciplinary Robot Engineering Systems, Hanyang University. Her research interest includes multi-robot systems, path planning for multiple mobile robots or agents, task and motion planning.
E-mail: st9051@hanyang.ac.kr

SeoHyun Yoo
M.Sc. candidate in the Department of Interdisciplinary Robot Engineering Systems, Hanyang University. Her research interest includes multi-robot systems, path planning for multiple mobile robots or agents, task and motion planning.
E-mail: sh0430@hanyang.ac.kr

HyoJae Kang
Ph.D. candidate in the Department of Interdisciplinary Robot Engineering Systems, Hanyang University. His research interest includes mechanism design, control architectures and programming, gripper and other end-effectors, and entertainment robotics.
E-mail: majaae5@hanyang.ac.kr

ChanHui Jung
Researcher in the Department of Robot Engineering, Hanyang University. His research interest includes precision engineering. multi-robot systems, path planning for multiple mobile robots or agents, task and motion planning.
E-mail: qhrejddlvltm@hanyang.ac.kr

Daehee Han
M.Sc. candidate in the Department of Interdisciplinary Robot Engineering Systems, Hanyang University. His research interest motion and path planning, integrated planning and control, AI-enabled robotics, and reinforcement learning.
E-mail: hdh1645@hanyang.ac.kr

Min-Sung Kang
Associate professor in the School of Smart Convergence Engineering, Hanyang University. His research interest includes mechanism design, planning, control architectures and programming, embedded systems.
E-mail: wowmecha@hanyang.ac.kr