Multi-Objective Multi-Agent Path Finding Based on Two-Stage Evolutionary Algorithms

Haruto Ando*, Yoshiki Nogata†, Tomohiro Harada†, Fumito Uwano‡
*Faculty of Engineering, Saitama University, Saitama, Japan
Email: ando.h.785@ms.saitama-u.ac.jp
†Graduate School of Science and Engineering, Saitama University, Saitama, Japan
Email: {nogata.y.564@ms., tharada@mail.}saitama-u.ac.jp
‡Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama, Japan
Email: uwano@okayama-u.ac.jp

***Abstract*—Multi-agent path finding (MAPF) is a critical problem in applications such as warehouse automation and autonomous driving. In real-world settings, multiple conflicting objectives, such as minimizing path length and ensuring safety, often must be optimized simultaneously, motivating the study of multi-objective MAPF (MOMAPF). However, existing MOMAPF approaches suffer from severe scalability limitations as the number of agents and the environment size increase. To address this challenge, we propose Two-Stage Evolutionary MOMAPF (TSE-MOMAPF), which optimizes a single agent's path via NSGA-II and refines the path set using an SMS-EMOA-based framework. Experiments on various grid maps show that the proposed method achieves superior scalability and provides broader Pareto front coverage in medium- and large-scale scenarios compared to an existing method. These results indicate that the proposed evolutionary framework effectively balances solution diversity and computational efficiency under strict time constraints.**

***Index Terms*—Multi-objective optimization, evolutionary algorithms, multi-objective multi-agent path finding, path planning**

I. Introduction

Multi-agent path finding (MAPF) is the problem of obtaining a set of collision-free paths that connect the respective start and goal positions of multiple agents [1], [2]. MAPF has attracted significant attention due to its wide range of applications, including warehouse automation using transport robots [3] and autonomous vehicle driving [4].

In real-world settings, MAPF often involves multiple conflicting objectives, such as minimizing path length and ensuring path safety. This study, therefore, focuses on multi-objective MAPF (MOMAPF) [5], which generalizes MAPF to handle multiple objectives. A notable approach for MOMAPF is multi-objective conflict-based search (MO-CBS) [6], which extends the conflict-based search (CBS) framework [7] from single-objective to a multi-objective optimization. However, it suffers from exponential growth in the search space, hindering efficiency in large-scale environments.

To address this challenge, we propose a novel method for solving MOMAPF using multi-objective evolutionary algorithms (MOEAs) [8]. Our method aims to overcome the inefficiencies of conventional approaches and to discover high-quality Pareto solutions within a realistic running time.

To investigate the effectiveness of the proposed method, we conduct experiments comparing it with MO-CBS under identical time constraints across four grid-based maps of varying sizes and numbers of agents.

This work was supported by JSPS Grant-in-Aid for Scientific Research (B) (Grant Number: 24K03001).

II. Problem Formulation

In MOMAPF, we define $A = \{1, 2, \ldots, N\}$ as a set of N agents placed in a 2D space. This 2D space is defined as a grid map G where cells are connected in four directions.

Each agent $i \in A$ has a start position s^i and a goal position g^i, and all agents start moving at time $t = 0$. An agent can move to one of the four neighboring grid cells in each time unit. $\pi^i\,(i \in \{1, 2, \ldots, N\})$ represents the path for agent i from s^i to g^i expressed as $\pi^i = \left\{s^i, p_1^i, p_2^i, \ldots, g^i\right\}$, where p_k^i represents the coordinates of agent i at time k. The set of paths for all agents, $\pi = \left\{\pi^1, \pi^2, \ldots, \pi^N\right\}$, is one solution for MOMAPF.

A. Constraints

A solution π must satisfy the constraint that no conflicts occur between any pair of agents $i \neq j$. Solutions with one or more conflicts are regarded as infeasible. Conflicts are defined by two patterns: vertex conflicts and edge conflicts. A vertex conflict occurs when two agents occupy the same coordinate p at the same time t. An edge conflict occurs when two agents pass through the same edge from opposite directions between time t and $t + 1$.

B. Objectives

The MOMAPF search considers M different objectives. Each objective is defined using a function $f_m\,(m \in \{1, 2, \ldots, M\})$, and the multi-objective function F to be minimized is defined as:

$$\text{minimize } F(\pi) = \left(f_1\left(\pi\right), f_2\left(\pi\right), \ldots, f_M\left(\pi\right)\right) \tag{1}$$

Since MOMAPF is a multi-objective optimization problem, solution dominance is defined by comparing their corresponding objective function vectors. Given two solutions π_a and π_b, along with their objective function vectors $F\left(\pi_a\right)$ and

$F(\pi_b)$, π_a is defined to dominate π_b (denoted as $\pi_a \prec \pi_b$) if $f_m(\pi_a) \leq f_m(\pi_b)$ for all objectives and $f_m(\pi_a) < f_m(\pi_b)$ for at least one objective. Conversely, if two solutions do not dominate each other, they are defined as non-dominated. The Pareto optimal solution set (POS) refers to the set of non-dominated solutions that are not dominated by any other solution among all feasible solutions Λ, which is defined as:

$$POS = \{\pi^* \in \Lambda \mid \forall\, \pi' \in \Lambda, \pi' \not\prec \pi^*\} \tag{2}$$

In the context of MOMAPF, the goal is to compute a diverse set of collision-free (i.e., feasible) Pareto-optimal path solutions within realistic time constraints.

While optimizing a single agent's path is common, MAPF requires solving this problem for multiple agents and then detecting and resolving conflicts. Furthermore, Pareto-optimal solutions must be considered at both the single- and multi-agent optimization levels. These characteristics make MOMAPF challenging for EAs or heuristics.

III. Previous Work: MO-CBS

MO-CBS [6] has been proposed as a conventional method for MOMAPF. MO-CBS is an algorithm that extends the CBS framework [7], which addresses single-objective MAPF, to multi-objective optimization. Similar to CBS, it is characterized by a two-stage search process: a low-level search and a high-level search. MO-CBS processes are described below.

A. Low-Level Search

In the low-level search, path planning is conducted for a single agent i. This path planning is subject to a constraint set Ω^i provided by the high-level search to avoid conflicts with other agents. From the paths that satisfy all constraints, a Pareto optimal path set $\Pi^i_* = \left\{\pi^i_a, \pi^i_b, \dots\right\}$ is calculated. For path generation, an existing multi-objective path planning algorithm for a single agent, such as NAMOA*-dr [9], is used. In particular, for two objectives, BOA* [10], an expedited version of NAMOA*-dr, is used.

B. High-Level Search

In the high-level search, a tree structure is used to resolve conflicts among agents using the CBS framework. Each node in the tree consists of $P = \{\pi, F(\pi), \Omega\}$. $F(\pi)$ represents the objective function value vector of solution π, and Ω represents the set of constraints imposed on this node. Each node is added to a priority queue $OPEN$, which is sorted lexicographically based on mutually non-dominated objective function value vectors $F(\pi)$. When executing the search, one node P is selected from the queue. If a conflict is found in the solution π of the selected node, constraints are generated for each agent to avoid the conflict. After splitting the node, a low-level search is performed on each resulting node to update π. If no conflict exists in the solution, the solution π and its objective function value vector $F(\pi)$ are added to the solution set $\mathcal{C}$.

C. Algorithm of MO-CBS

In MO-CBS, the initial path set Π^i_0 for agent i is generated using the algorithms employed in the low-level search (NAMOA*-dr or BOA*). An initial solution set $\Pi_0 = \left\{\pi_0 | \pi_0 = \left(\pi^1_0, \pi^2_0, \dots, \pi^N_0\right), \pi^i_0 \in \Pi^i_0\right\}$ is then created by considering all possible combinations of paths from the initial path sets of all agents. For each of these solutions $\pi_0 \in \Pi_0$, a node P is generated and added to $OPEN$. This node contains the corresponding objective function value vector $F(\pi)$ and an empty constraint set $\Omega = \emptyset$.

After initializing $OPEN$, the high-level and low-level searches are repeated alternately until $OPEN$ becomes empty. When adding solution π and its objective function value vector $F(\pi)$ to the solution set $\mathcal{C}$ during the high-level search, any objective function value vectors within $\mathcal{C}$ that are dominated by $F(\pi)$ are removed. Consequently, only non-dominated solutions remain in the solution set $\mathcal{C}$ at the end of the search. Conversely, if a solution that dominates the objective function vector $F(\pi)$ already exists in the solution set $\mathcal{C}$, it is not added. This ensures that the solution set $\mathcal{C}$ becomes the Pareto optimal solution set upon completion of the search.

IV. Proposed Method

MO-CBS becomes inefficient as the problem scale increases, for example, in the number of agents or the environment size. To address this issue, this study proposes a novel approach for solving MOMAPF, named Two-Stage Evolutionary MOMAPF (TSE-MOMAPF), that ensures scalability for large-scale problems by incorporating MOEAs.

The proposed method consists of two stages: the first optimizes the path of a single agent, π^i, and the second optimizes the set of paths, π. This approach is motivated by the complexity of simultaneously optimizing the paths of all agents. Instead, our method refines the path set π by modifying the path of a single agent, π^i, in a manner similar to a mutation process. The processes of the algorithm are described below.

A. Initialization

At the beginning of TSE-MOMAPF, each agent's path set, Π^i, is initialized. The initialization process is shown in Alg. 1.

First, the minimum-cost path for each objective m is computed backward from the goal point g^i using Dijkstra's algorithm to determine the minimum-cost path from any location on the map to g^i. The map area used for this search is restricted to a rectangle with vertices at the start point s^i and the goal point g^i of agent i, expanded by a fixed distance D. This improves the efficiency of the initialization process for large-scale problems.

Subsequently, a path is constructed from the start point s^i to the goal point g^i by following the minimum cost path π^i_m. During this process, randomness is introduced by selecting paths other than the minimum cost path, including backtracking paths, with a constant probability P_{dev}. This approach is inspired by the algorithm proposed by Eppstein [11] and ensures both path quality and randomness.

Algorithm 1 Pseudo-code of single agent path set initialization

Input: s^i, g^i, D, P_{dev}
Output: Π^i

1: $G' \leftarrow$ Area in G limited by s^i, g^i, and D
2: **for** Each objective m **do**
3: Calculate minimum cost path from $v' \in G'$ to g^i
4: $\pi^i_m \leftarrow$ Minimum cost path from s^i to g^i
5: **for** $\lfloor N_s/M \rfloor$ or $\lceil N_s/M \rceil$ **do**
6: $\pi^i \leftarrow$ Deviate from π^i_m with P_{dev}
7: $\Pi^i \leftarrow \Pi^i \cup \{\pi^i\}$
8: **end for**
9: **end for**
10: **return** Π^i

For each objective, either $\lfloor N_s/M \rfloor$ or $\lceil N_s/M \rceil$ paths are generated such that $\left|\Pi^i\right| = N_s$. The decision of which number of paths to generate for each objective is made randomly.

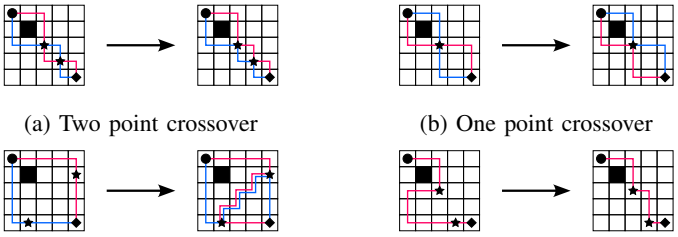
B. Optimizing Single Agent Path

NSGA-II, a widely used MOEA for bi- or three-objective optimizations, is employed to optimize the path π^i of a single agent. This single-agent optimization is employed during initial solution generation and as a mutation operator to optimize multi-agent path sets (as described in Section IV-C).

1) Crossover: The crossover between two paths depends on the number of locations at which agents pass through the same coordinate p at the same time t. Fig. 1a through Fig. 1c illustrate these crossover methods: the start and end points of the paths are represented by circles and squares, respectively, while the aforementioned locations are indicated by star symbols. If there are two or more shared locations, two points are randomly selected, and a two-point crossover is applied as shown in Fig. 1a. If there is only one shared location, a one-point crossover is used as shown in Fig. 1b. If there are no shared locations, one random point is chosen from each path, and a path synthesis crossover is performed to connect these two points as shown in Fig. 1c. The connecting path is generated to minimize the cost of a randomly selected single objective. Fig. 1c shows the resulting path when the path length is minimized.

2) Mutation: The mutation involves selecting two random points in the path and mutating the segment between them. The new path segment is generated to minimize the cost of a randomly selected single objective. Fig. 1d illustrates an example of the mutation, where two randomly selected points are indicated by star symbols and connected so that the path length is minimized.

3) Algorithm of Single Agent Optimization: Alg. 2 shows the optimization algorithm for a single agent path π^i. The optimization loop from line 2 to line 5 is repeated for a specified number of generations, as per the NSGA-II framework. For the offspring generation in line 3, crossover and mutation are employed. As in NSGA-II, both parent selection in line 3 and survival selection in line 4 use non-dominated sorting and crowding distance. Finally, in line 7, non-dominated



(a) Two point crossover (b) One point crossover

(c) Path synthesis crossover (d) An example of mutation

Fig. 1: Examples of crossovers and mutation

Algorithm 2 Pseudo-code of single agent path optimization

Input: Π^i
Output: $\mathcal{S}^i, \Pi^i$

1: $\mathcal{P} \leftarrow \Pi^i$ ▷ Inherit initial path set
2: **repeat**
3: $\mathcal{Q} \leftarrow$ Generate offspring from $\mathcal{P}$ using crossover and mutation
4: $\mathcal{P} \leftarrow$ Select next population from $\mathcal{P} \cup \mathcal{Q}$ using non-dominated sorting and crowding distance
5: **until** Termination condition is met
6: $\Pi^i \leftarrow \mathcal{P}$
7: $\mathcal{S}^i \leftarrow$ Non-dominated solutions in $\mathcal{P}$
8: **return** $\mathcal{S}^i, \Pi^i$

solutions are selected from the final population $\mathcal{P}$ to form the algorithm's output solution set $\mathcal{S}^i$.

C. Optimizing Multiple Agent Path Set

The optimization of the path set π is inspired by SMS-EMOA [12]. We employ SMS-EMOA because its steady-state update structure—where a fixed-size population is maintained by incrementally incorporating new offspring—naturally fits our method, in which modifying a single agent's path yields a variable number of new candidate solutions that must be selectively merged into the population based on HV contribution. The individual processes are described below.

1) Initialization: After executing Alg. 1 for all agents and initializing the path sets Π^i, the solution set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{N_m}\}$, which combines the paths of all agents, is initialized. Each solution $\pi_k = \{\pi^1_k, \pi^2_k, \ldots, \pi^N_k | \pi^i_k \in \Pi^i\}$ is constructed by combining randomly selected paths, one for each agent, from its corresponding generated path sets Π^i. After selecting the paths, the conflicts among agents is checked. If any conflicts exist, the solution is discarded, and new combinations are generated until $|\Pi| = N_m$ is satisfied.

2) Offspring Generation: One solution $\pi_k = \{\pi^1_k, \pi^2_k, \ldots, \pi^N_k\}$ is randomly selected from the population $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{N_m}\}$ and used as a parent to generate offspring. An agent $i \in A$ is chosen from the parent π_k, and the single agent path optimization in Alg. 2 is executed for this agent. To improve search efficiency, the existing path set Π^i for agent i is utilized as the initial population for this

optimization. This process results in an updated path set Π^i and its non-dominated set $\mathcal{S}^i$.

Next, each path $\pi^i_{new} \in \mathcal{S}^i$ is used to create offspring $\pi_{new} = (\pi_k \setminus \{\pi^i_k\}) \cup \{\pi^i_{new}\}$ by replacing the path of agent i. If π_{new} contains any conflicts, it is excluded; otherwise, it is added to the candidate population $\mathcal{P}'$ for the next generation.

3) Survival Selection: The next generation Π_{new} is selected from the candidate population $\mathcal{P}'$ such that $|\Pi_{new}| = N_m$.

First, non-dominated sorting is performed on the population $\mathcal{P}'$ to classify individuals into several ranks $\left\{\mathcal{F}_1, \mathcal{F}_2, \dots \mid (\pi_{(a)}, \pi_{(b)}, \dots) \in \mathcal{F}_k\right\}$ based on Pareto optimality. Individuals are added to the next generation Π_{new} starting from the best rank $\mathcal{F}_1$. If adding the entire population of rank $\mathcal{F}_k$ results in $|\Pi_{new}| > N_m$, individuals are selected from $\mathcal{F}_k$ until the condition $|\Pi_{new}| = N_m$ is met. For the population truncation, the hypervolume (HV) contribution is used as the selection criterion. This metric quantifies each individual's contribution to the total HV of the Pareto front. When calculating HV contribution, the objective function values are normalized to the range $[0, 1]$, and the reference point is set to 1.1 for all objectives. Individuals are added to Π_{new} one by one, starting with the one that exhibits the largest contribution. The new population Π_{new} is used as the next population Π. Finally, the non-dominated solution set $\mathcal{S}$ serves as the final solution of the algorithm.

4) Algorithm of Multi-agent Optimization: Alg. 3 shows the optimization algorithm for solution π. Lines 1 and 2 describe the initialization of the path set Π^i for each agent, and the solution set Π. From lines 3 to 13, offspring π_{new} having no conflicts are generated and added to the population $\mathcal{P}'$. Lines 14 to 25 involve the survival selection using non-dominated sorting and HV contribution.

Algorithm 3 Pseudo-code of multi-agent path optimization

Output: Optimized non-dominated solutions $\mathcal{S}$
1: $\Pi^i \leftarrow$Initialize each agent's path using Alg. 1
2: $\Pi \leftarrow$Initialize population by randomly combining $\pi^i \in \Pi^i$
3: **repeat**
4: Choose random path set $\pi_k \in \Pi$
5: Choose random agent path $\pi^i_k \in \pi_k$
6: $\mathcal{P}' \leftarrow \Pi$
7: $\mathcal{S}^i, \Pi^i \leftarrow$ Optimize single agent π^i using Alg. 2 initialized with Π^i
8: **for all** $\pi^i_{new} \in \mathcal{S}^i$ **do**
9: $\pi_{new} \leftarrow \left(\pi_k \setminus \left\{\pi^i_k\right\}\right) \cup \left\{\pi^i_{new}\right\}$
10: **if** π does not have conflict **then**
11: $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\pi_{new}\}$
12: **end if**
13: **end for**
14: $\mathcal{F}_1, \mathcal{F}_2, \dots \leftarrow$ Non-dominated sort$(\mathcal{P}')$
15: $\Pi_{new} \leftarrow \emptyset$
16: **for all** $\mathcal{F}_i \in \{\mathcal{F}_1, \mathcal{F}_2, \dots\}$ **do**
17: **if** $|\mathcal{F}_i \cup \Pi_{new}| \leq N_m$ **then**
18: $\Pi_{new} \leftarrow \mathcal{F}_i \cup \Pi_{new}$
19: **else**
20: Calculate HV contributions for all solutions in $\mathcal{F}_i$
21: $\mathcal{F}'_i \leftarrow$ Select solutions having top $N_m - |\Pi_{new}|$ HV contributions
22: $\Pi_{new} \leftarrow \Pi_{new} \cup \mathcal{F}'_i$
23: **end if**
24: **end for**
25: $\Pi \leftarrow \Pi_{new}$
26: $\mathcal{S} \leftarrow$ Non-dominated solutions in $\Pi \cup \mathcal{S}$
27: **until** Termination condition is met
28: **return** $\mathcal{S}$

V. Experiment

To evaluate the effectiveness of TSE-MOMAPF, experiments are conducted to compare it with MO-CBS.

A. Problem Settings

The experiment focuses on MOMAPF on a grid map G. The grid maps G and the start, and goal points $\left(s^i, g^i\right)$ are obtained from the "random" category of the existing MAPF dataset [2]. In this dataset, obstacles on the map are placed on each grid cell with a 10% probability, and the start and goal points for each agent are generated by randomly selecting reachable locations on the map. Three map sizes are used for map G: 32×32 grids, 64×64 grids, and 96×96 grids. These maps are subsequently referred to as G32, G64, and G96. For G32 and G64, the existing datasets are utilized. On the other hand, since a G96 dataset is not provided, it is generated according to the previous study [2]. The number of agents N is set to $N = 10$ for G32 and G96, while two settings, $N = 10$ and $N = 15$, are considered for G64.

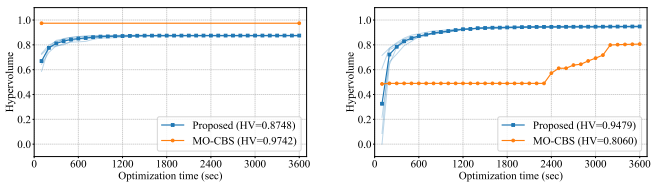
In this experiment, two objectives are considered: path length (Efficiency) and path cost (Safety). The path length is the sum of the time steps t required for all agents to reach their goals, and the path cost is the total risk of the grids each agent traverses. The risk of each grid is calculated as the sum of the number of obstacles in the eight surrounding grids. This setting is identical to the previous study [6].

B. Parameters

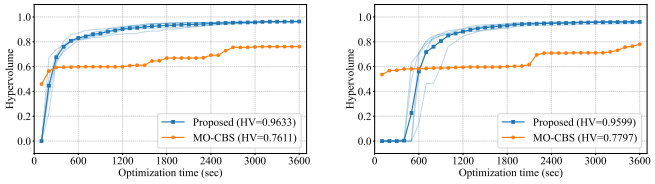
The number of path sets for each agent in the first stage optimization is set to $N_s = 30$, and the number of path sets for all agents in the second stage optimization is $N_m = 50$. The search range during initialization is $D = 3$, and the probability for initial path generation is $P_{dev} = 0.2$. The crossover and mutation rates of NSGA-II are set to 0.9 and 0.1, respectively.

C. Evaluation Criteria

To evaluate both methods fairly, the execution time is fixed at 3,600 seconds, and the resulting final solution sets are evaluated using HV. For the proposed method, the HV values are the average of 5 trials, whereas a single trial is conducted for MO-CBS because it is a deterministic method. Each objective value is normalized to the range $[0, 1]$ using the maximum and minimum values observed in the non-dominated solution sets across all generations in each trial. Finally, the HV is calculated using a reference point of $(1.1, 1.1)$.

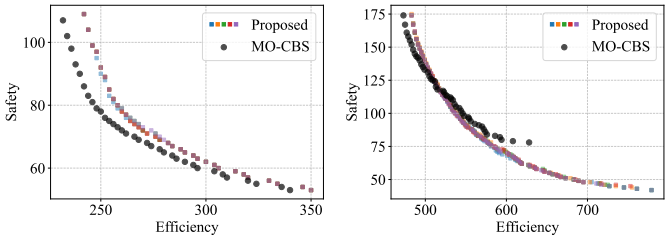


(a) G32, $N = 10$ (b) G64, $N = 10$

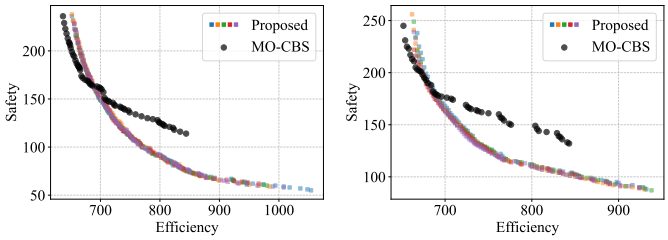


(c) G64, $N = 15$ (d) G96, $N = 10$

Fig. 2: Transition of HV calculated at 100-second intervals



(a) G32, $N = 10$ (b) G64, $N = 10$



(c) G64, $N = 15$ (d) G96, $N = 10$

Fig. 3: Non-dominated solution set obtained by two methods

VI. Result

Fig. 2 shows the transitions of the HV values for the obtained non-dominated solution sets, calculated at 100-second intervals. The numerical values in the legend indicate the HV values for each method at 3,600 seconds. For the proposed method, the HV values obtained in each trial are represented by thin lines, while their average is shown as a bold line.

As shown in Fig. 2, MO-CBS terminates within 100 seconds and finds POS for the small-scale G32 map. Throughout the entire 3,600-second time horizon, MO-CBS consistently maintains a higher HV value than the proposed method.

In contrast, for the larger G64 and G96 maps, the proposed method surpasses MO-CBS in HV early in the search and outperforms it thereafter. Although MO-CBS initially achieves relatively high HV values, both algorithms exhibit HV values that are lower than their final results in the early stage, as the search space has not yet been sufficiently explored.

Furthermore, the proposed method attains HV values comparable to its final performance from the early phases of optimization. These results indicate that the proposed method delivers solutions with stable and consistent quality throughout the search process.

Fig. 3 shows the final non-dominated solution sets. The horizontal axis indicates the path length (Efficiency), and the vertical axis indicates the path cost (Safety). The results of the proposed method from five trials are shown as colored points, while those of MO-CBS are shown as black points.

As shown in Fig. 3, MO-CBS finds a better Pareto front than the proposed method in G32. On the other hand, MO-CBS failed to obtain the entire Pareto front in G64 and G96. In contrast, the proposed method finds solutions over a wider range of the objective space than MO-CBS does in G64 and G96. In particular, the proposed method finds safer but less efficient paths in the lower-right region that MO-CBS does not find. This result stems from the fact that MO-CBS performs the search in lexicographical order, which causes optimization to proceed from the region where the path length is minimized. This difference becomes more pronounced as the problem scale increases. In G64, the difference is larger for $N = 15$ than for $N = 10$, and, for the same number of agents, it is larger in G96 than in G64. These results demonstrate the high scalability of the proposed method for large-scale problems.

Fig. 4 shows the paths of each agent for three solutions obtained by the proposed method in a scenario with the G64 map and $N = 10$ agents (Fig. 3b): specifically, the solutions with the minimum path length, the minimum path cost, and an intermediate solution are illustrated, respectively. In Fig. 4, black cells indicate the obstacles.

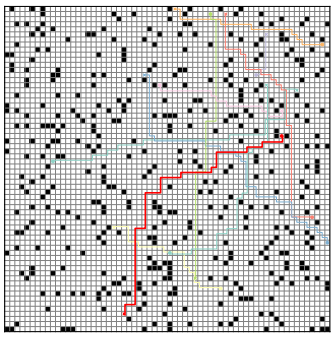
From Fig. 4, we see that the proposed method finds paths for each objective. In the solution where path length is minimized (Fig. 4a), agents follow the shortest path even if it involves grids near high-cost obstacles. In contrast, in the solution where path cost is minimized (Fig. 4c), agents take routes that detour around high-cost grids. For example, the agent indicated by the red line in Fig. 4a follows a linear path by accepting an increase in path cost when the path length is minimized. Conversely, when the path cost is minimized, the agent indicated by the red line in Fig. 4c follows a significantly longer detouring route. The intermediate solution (Fig. 4b) lies between the two. As a result, the balance between path length and path cost varies across solutions.

VII. Discussion

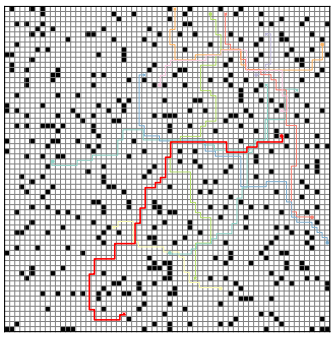
Experimental results revealed that the proposed method achieves high performance in obtaining a comprehensive Pareto front within a fixed runtime, even for large-scale problems. The factors contributing to these results are discussed from two perspectives: scalability and solution diversity.

A. High Scalability

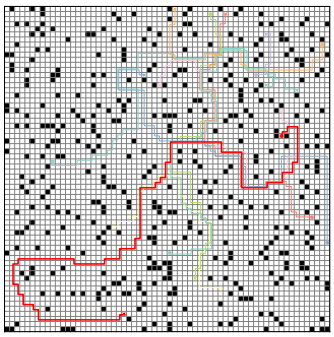
MO-CBS showed a tendency for solution quality to decrease compared to the proposed method as the problem size



(a) Minimum path length solution
(Efficiently: 66, Safety: 20)



(b) Intermediate solution
(Efficiently: 86, Safety: 10)



(c) Minimum path cost solution
(Efficiently: 142, Safety: 5)

Fig. 4: Example of path sets obtained by the proposed method in G64 and $N = 10$

increased. This is due to the structure of MO-CBS, which explicitly detects and resolves conflicts among agents during high-level search. Specifically, as the number of agents N increases, the number of conflicts to resolve grows, leading to a significantly larger search tree.

In contrast, the proposed method is based on EAs and probabilistically searches for promising regions. It also includes processes to reduce the problem scale, such as limiting the search range during initialization and removing infeasible solutions that contain conflicts. These structural differences make the algorithm computationally efficient even as the problem scale grows, contributing to high scalability.

B. Diversity of Solutions

The solution set $\mathcal{S}$ produced by the proposed method yields a more comprehensive Pareto front in the search space than MO-CBS. This result stems from the fact that MO-CBS searches in lexicographical order of costs, leading to biased solutions during execution and preventing the algorithm from covering the entire Pareto front. This bias in MO-CBS is evident in the results for G64 and G96 in Fig. 3.

In contrast, the proposed method is based on EAs and explores the entire objective space simultaneously through a population-based search. As a result, it maintains a set of solutions that cover the entire Pareto front at any stage of the algorithm, allowing it to provide solutions across a wide range of the objective space within any given time. This difference helps ensure solution diversity within a limited running time.

VIII. Conclusion

This study proposed a novel method, TSE-MOMAPF, for solving large-scale MOMAPF problems within a realistic time frame. Experimental results demonstrated that the proposed method obtains higher-quality Pareto solutions under time constraints than MO-CBS. Specifically, the proposed method demonstrated strong scalability for large-scale problems, such as those with more agents or a larger map.

Future work will involve improving the proposed method to further enhance scalability. For example, while the current approach removes infeasible solutions that contain conflicts, we will investigate techniques to utilize these solutions to reach high-quality feasible solutions.

Acknowledgements

During the preparation of this work, the authors used AI systems to improve language and readability. After using these tools, the authors reviewed and edited the content as needed and took full responsibility for the publication.

References

[1] J.-M. Alkazzi and K. Okumura, "A comprehensive review on leveraging machine learning for multi-agent path finding," *IEEE Access*, vol. 12, pp. 57 390–57 409, 2024.

[2] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. Kumar, R. Barták, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, no. 1, pp. 151–158, Sep. 2021.

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

[4] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[5] J. Weise, S. Mai, H. Zille, and S. Mostaghim, "On the scalable multi-objective multi-agent pathfinding problem," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[6] Z. Ren, S. Rathinam, and H. Choset, "A conflict-based search framework for multiobjective multiagent path finding," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1262–1274, 2023.

[7] 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.

[8] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Arch. Comput. Methods Eng.*, vol. 29, no. 7, pp. 5605–5633, Nov. 2022.

[9] F.-J. Pulido, L. Mandow, and J.-L. Pérez-de-la Cruz, "Dimensionality reduction in multiobjective shortest path search," *Computers & Operations Research*, vol. 64, pp. 60–70, 2015.

[10] C. H. Ulloa, W. Yeoh, J. A. Baier, H. Zhang, L. Suazo, and S. Koenig, "A simple and fast bi-objective search algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 143–151.

[11] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652–673, 1998.

[12] N. Beume, B. Naujoks, and M. Emmerich, "Sms-emoa: Multiobjective selection based on dominated hypervolume," *European journal of operational research*, vol. 181, no. 3, pp. 1653–1669, 2007.