Multi-Objective A* Algorithm for the Multimodal Multi-Objective Path Planning Optimization

Bo Jin
College of Management
Shenzhen University
Shenzhen, China
jinbo@szu.edu.cn

***Abstract*—In this paper, we consider the multimodal multi-objective path planning (MMOPP) optimization, which is the main topic of a special session in IEEE CEC 2021. The MMOPP aims at finding all the Pareto optimal paths from a start area to a goal area on a grid map, while passing through several designated must-visit areas. We propose an efficient approach based on the multi-objective A* algorithm to exactly solve the MMOPP. Experiments are conducted on the official MMOPP test suite to evaluate the performance of the proposed approach. We also show the admissibility of the proposed approach so that the computational results can be used as the standard answers to the MMOPP test suite.**

***Index Terms*—Combinatorial optimization, multimodal multi-objective path planning optimization, multi-objective A* algorithm, best-first search, informed search.**

I. Introduction

Multi-objective decision-making is common in a variety of real-world applications. A multi-objective optimization problem deals with a number of objectives that are usually in conflict with each other. In most cases, it is impossible to reach the best values for all individual objectives in one single solution. Therefore, the goal of multi-objective optimization is to find as many trade-off solutions as possible, so that a decision maker would be in a better position to make a reasonable choice among these unveiled alternatives [1].

Suppose that the multi-objective optimization problem under study is stated as follows:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned}$$

where we have M real-valued objective functions $f_1, \ldots, f_M$ to be minimized simultaneously. The image of the *decision space* $\mathcal{X}$ under the mapping $\boldsymbol{f} = (f_1, \ldots, f_M)$ is referred to as the *objective space* and denoted by $\mathcal{Y}$. Thus, every feasible solution $\boldsymbol{x} \in \mathcal{X}$ corresponds to an M-dimensional objective vector $\boldsymbol{f}(\boldsymbol{x}) \in \mathcal{Y}$. A vector $\boldsymbol{y} = (y_1, \ldots, y_M)$ is said to *dominate* another vector $\boldsymbol{y}' = (y'_1, \ldots, y'_M)$ if $y_i \leq y'_i$ for all $i \in \{1, \ldots, M\}$ and $y_j < y'_j$ for some $j \in \{1, \ldots, M\}$. A feasible solution $\boldsymbol{x}$ is said to *dominate* another feasible solution $\boldsymbol{x}'$ with respect to $\mathcal{X}$ if $\boldsymbol{f}(\boldsymbol{x})$ dominates $\boldsymbol{f}(\boldsymbol{x}')$. A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is said to be *Pareto optimal* if it is not dominated by any other feasible solutions in $\mathcal{X}$. The *Pareto set*, which we denote by $\mathcal{X}^*$, is defined as the set of all Pareto optimal solutions in the decision space, and the corresponding points in the objective space constitute the *Pareto front*, which we denote by $\mathcal{Y}^*$.

In the past decades, evolutionary algorithms have become the mainstream for solving multi-objective optimization problems, due to the adoption of the population concept, which allows multiple solutions to be searched simultaneously in one single simulation, e.g., NSGA-II [2], SPEA2 [3], MOEA/D [4], etc. However, most existing multi-objective evolutionary algorithms only aim at approximating the Pareto front, but neglect the distribution of solutions in the decision space [5]. As stated above, multiple available Pareto optimal solutions support a reliable decision-making process. If these solutions have great diversity in the decision space, they can further provide more insightful information to the decision makers [6]. Thus, a good approximation to both Pareto front and Pareto set should be required by the decision maker for discovering hidden properties and facilitating the decision-making. Optimization problems with this specific goal are referred to as *multimodal* multi-objective optimization. For a comprehensive review on multimodal multi-objective optimization, please refer to a recent survey by Tanabe and Ishibuchi [7].

Multi-objective path planning is a typical optimization problem raised in various fields such as motion planning [8], urban transportation [9], and vehicle routing [10]. Generally, a single objective function is not sufficient to characterize practical path planning problems. Various metrics (e.g., path length, travel time, total cost, and gas emission) may all need to be optimized, yet they are usually conflicting and incommensurate in real-world situations. This study focuses on a new challenge to the subject of multi-objetive path planning, termed the *multimodal multi-objective path planning* (MMOPP) optimization, which is the main topic of a special session in IEEE CEC 2021. In few words, the MMOPP involves finding all the Pareto optimal paths from a start area to a goal area on a grid map, while these paths are required to pass through several designated must-visit areas. The special session is also accompanied by a competition and releases an MMOPP test suite to the participants. The test suite contains 12 test problems with different characteristics, simulating the goals and constraints of real-life path planning.

Clearly, the MMOPP is a particular extension of the multi-objective shortest path problem. In the following, we review a few representative exact algorithms for the multi-objective

shortest path problem, which motivate the proposed approach in our work. Hansen [11] first extended Dijkstra's algorithm [12] for the bi-objective shortest path problem, and Martins [13] further extended it to the multi-objective case. When the goal node is specified in the problem, an informed search (e.g., A* [14]) accepts heuristic information to improve the search efficiency. This information is obtained by a heuristic function that estimates how close a node is to the goal node. There have been several studies proposing multi-objective A* (MOA*) algorithms. Stewart and White [15] outlined the first multi-objective extension of A*, which we refer to as SW-MOA* in this paper to avoid confusions. Mandow and Pérez de la Cruz [16] proposed a new algorithm called NAMOA*, where "NA" stands for "new approach". They showed a reduction in the space requirements of NAMOA* over those of SW-MOA*. A more in-depth comparison can be found in [17]. Pulido et al. [18] described a new dimensionality reduction technique called t-discarding to speed up dominance checks, and applied it to NAMOA* to achieve a reduction in time requirements.

The multi-objective shortest path problem is already intractable, since the number of distinct non-dominated paths grows exponentially with the graph size in the worst case, even for the bi-objective case [11]. On this basis, the MMOPP in addition involves several must-visit nodes, which further increase the complexity of the problem. The purpose of this study is to develop an efficient MOA*-based approach to exactly solve the MMOPP. We also show the admissibility of the proposed approach so that the computational results can be used as the standard answers to the MMOPP test suite.

The remainder of this paper is organized as follows. Section II gives the problem definition. Section III presents the proposed approach in details. Section IV demonstrates the computational results. Lastly, Section V concludes the paper.

II. Problem Description

In this section, we establish some useful notation and terminology to formally describe the MMOPP, in accordance with the document provided by the organizers of the special session [19].

The MMOPP is defined on a grid map, where the coordinate (x, y) refers to the square area located at the x-th column from the left and the y-th row from the top. A binary *map* matrix is used to define the original grid map: $map[x][y] = 0$ means that area (x, y) is passable, and $map[x][y] = 1$ means the opposite. A path is a sequence of passable areas where consecutive areas in the sequence are also adjacent in the map.

Each passable area (x, y) in the map is associated with an M-dimensional non-negative cost vector denoted by $cost[x][y]$. The cost of a path is also M-dimensional, which accumulates the costs of all the areas passed through. Each dimension of the cost vectors corresponds to a certain real-valued objective function to be minimized, e.g., the length of the path, the level of traffic congestion, the number of intersections encountered, and other preference indicators that people pay attention to when choosing a path.

A traveler needs to travel from a start area $(x^{\text{start}}, y^{\text{start}})$ to a goal area $(x^{\text{goal}}, y^{\text{goal}})$, while K key areas $(x_1^{\text{key}}, y_1^{\text{key}})$, …, $(x_K^{\text{key}}, y_K^{\text{key}})$ (i.e., must-visit areas) are required to be visited in an arbitrary order. Without loss of generality, the key areas are assumed to be distinct passable areas and different from the start area and the goal area. Moreover, the start area, the goal area, and the key areas are sometimes collectively referred to as the mandatory areas hereinafter in this paper.

Fig. 1 gives an example map, where the start area is colored in blue, the goal area is colored in green, the key areas are colored in yellow, and the other passable areas are colored in black. Fig. 2 exemplifies a feasible solution path in the map, which starts from the start area, passes through all the key areas, and finally reaches the goal area. The MMOPP aims at finding all the Pareto optimal solution paths whose cost vectors are not dominated by any other feasible solution paths in the map.

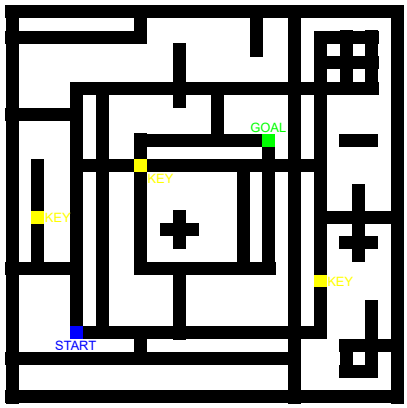

Fig. 1. An example map of the MMOPP.

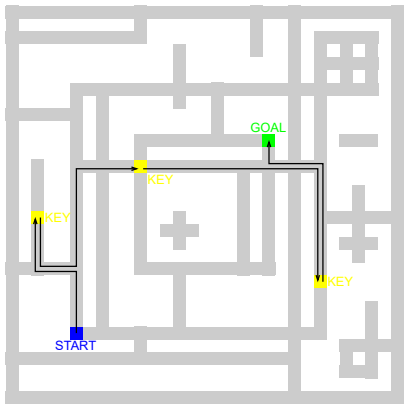

Fig. 2. An example of a solution path.

III. Proposed Approach

A. Feasibility Check and Graph modeling

Before formally solving the MMOPP, we need to first verify the feasibility of the problem. If the mandatory areas are not reachable from each other, then there is no feasible solution to the considered problem. This verification can be done in

polynomial time by analyzing the connected components of the map.

Besides, unnecessary areas can be removed from the map to potentially reduce the scale of the problem. We consider two types of map reduction. First, unreachable areas from the mandatory areas are useless in the path planning. Second, we investigate all the articulation points in the map. If a group of connected areas separated by an articulation point contains no mandatory area, we can simply omit this group.

The feasibility check and the map reduction are both accomplished by the procedure described in Algorithm 1. Remark that the procedure is an adaptation of Tarjan's algorithm [20] for detecting connected components and articulation points. The procedure first iterates over all the reachable areas from the start area by constructing a depth-first search (DFS) spanning tree. Relevant auxiliary matrices are introduced as follows:

- $depth[x][y]$ is the depth of area (x, y) in the DFS spanning tree, and $depth[x][y] = 0$ means that area (x, y) has not been visited by the DFS;
- $low[x][y]$ is the depth of the farthest ancestor that is adjacent to any area in the subtree rooted at area (x, y);
- $flag[x][y]$ is a boolean value that indicates whether the subtree rooted at area (x, y) contains any mandatory areas;
- $children[x][y]$ is the list of child nodes of area (x, y) in the DFS spanning tree;
- $retained[x][y]$ is a boolean value that indicates whether area (x, y) will be retained in the reduced map.

Algorithm 1. Feasibility check and map reduction.

local *depth*: Integer matrix initially filled with 0;
local *low*: Integer matrix initially filled with 0;
local *flag*: Boolean matrix initially filled with `false`;
local *children*: List matrix initially filled with empty lists;
global *retained*: Boolean matrix initially filled with `false`;

1: BUILDTREE($x^{\text{start}}, y^{\text{start}}, 1$);
2: TRIMTREE($x^{\text{start}}, y^{\text{start}}$);
3: **procedure** BUILDTREE(x, y, d) ▷ Recursive
4: $depth[x][y] \leftarrow d$;
5: $low[x][y] \leftarrow d$;
6: **if** $(x, y) = (x^{\text{start}}, y^{\text{start}})$ **or** $(x, y) = (x^{\text{goal}}, y^{\text{goal}})$ **or** $(x, y) \in \{(x_1^{\text{key}}, y_1^{\text{key}}), \ldots, (x_K^{\text{key}}, y_K^{\text{key}})\}$ **then**
7: $flag[x][y] \leftarrow$ `true`;
8: **else**
9: $flag[x][y] \leftarrow$ `false`;
10: **end if**
11: **for** passable area (x', y') adjacent to area (x, y) **do**
12: **if** $depth[x'][y'] = 0$ **then**
13: Append (x', y') to $children[x][y]$;
14: BUILDTREE($x', y', d + 1$);
15: $low[x][y] \leftarrow \min(low[x][y], low[x'][y'])$;
16: **if** $flag[x'][y']$ **then**
17: $flag[x][y] \leftarrow$ `true`;
18: **end if**
19: **else**
20: $low[x][y] \leftarrow \min(low[x][y], depth[x'][y'])$;
21: **end if**
22: **end for**
23: **end procedure**
24: **procedure** TRIMTREE(x, y) ▷ Recursive
25: $retained[x][y] \leftarrow$ `true`;
26: **for** $(x', y') \in children[x][y]$ **do**
27: **if** $low[x'][y'] < depth[x][y]$ **or** $flag[x'][y']$ **then**
28: TRIMTREE(x', y');
29: **end if**
30: **end for**
31: **end procedure**

Then, unnecessary areas are identified and removed from the map by another traversal on the DFS spanning tree. Consider an area (x, y) in the DFS spanning tree: if any of its child node $(x', y') \in children[x][y]$ satisfies $low[x'][y'] \geq depth[x][y]$ and $flag[x'][y'] =$ `false`, then the entire subtree rooted at (x', y') can be pruned.

The finally obtained matrix, denoted by *retained*, defines the reduced map. Thus, the feasibility of the MMOPP problem can be verified by checking whether all the mandatory areas are retained in the reduced map. Meantime, all the unnecessary areas that will never appear in a Pareto optimal solution have been removed to reduce the scale of the problem.

Fig. 3 shows the result of reducing the original map from Fig. 1. The unnecessary areas that have been removed are colored in gray. The degree of a retained area is defined as the number of adjacent retained areas in the reduced map. It is not difficult to verify that there is no retained area of degree 1 in the reduced map unless it is a mandatory area.

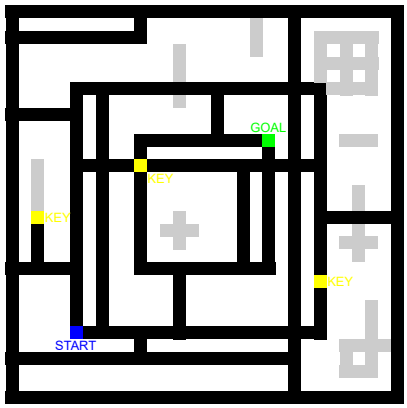

Fig. 3. Reduced map by removing unnecessary areas.

Based on the reduced map, we model the MMOPP into an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ as follows, where $\mathcal{N}$ is the node set and $\mathcal{E}$ is the edge set. Each node $n \in \mathcal{N}$ corresponds to a mandatory area or a retained area of degree 3 or 4, and each edge $e \in \mathcal{E}$ corresponds to a sequence of continuous retained areas of degree 2 between two nodes. For each node $n \in \mathcal{N}$, let (x_n, y_n) denote the coordinate of node n, and Δ_n the set of edges incident with node n. Specially, let n^{start}, n^{goal}, and $n_1^{\text{key}}, \ldots, n_K^{\text{key}}$ denote the start node, the goal node, and the key nodes, respectively. For each edge $e \in \Delta_n$ incident with node $n \in \mathcal{N}$, let $\delta_{n,e}$ denote the other end of edge e, ω_e the cumulative cost of all the intermediate areas in edge e, and $\Omega_{n,e}$ the sequence of intermediate areas from node n to node $\delta_{n,e}$. In this way, the MMOPP is transformed into a special

multi-objective shortest path problem with several designated must-visit nodes on an undirected graph.

The purpose of contracting continuous areas of degree 2 into one single edge is to further reduce the scale of the problem. Fig. 4 shows the undirected graph built based on the reduced map from Fig. 3. In this figure, each circle represents a node, and each thick line linking two nodes represents an edge. It is worth noting that there might be parallel edges between two nodes in the graph.

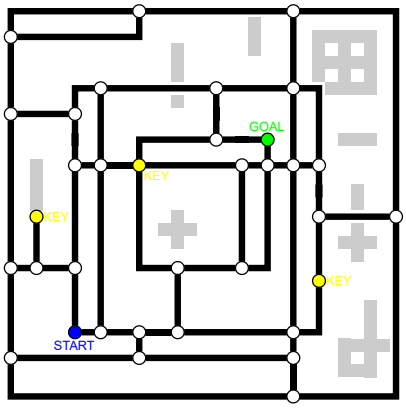

Fig. 4. Graph model based on the reduced map.

B. Multi-Objective A* Algorithm

The core of the proposed approach is an efficient multi-objective A* (MOA*) algorithm specially designed for the MMOPP optimization. The overall process of the proposed MOA* algorithm is similar to that of NAMOA* [16], with many differences in details. Next, we first introduce some fundamental concepts required for describing the proposed MOA* algorithm.

The term "status" is used to describe which key areas have been visited along a path. A status is expressed by a K-dimensional bit array or equivalently an integer from 0 to $2^K - 1$, where the k-th bit being 1 means that the k-th key area has been visited, otherwise the k-th key area has not been visited yet. Let $\mathcal{S} = \{0, \ldots, 2^K - 1\}$ denote the set of all the 2^K possible statuses, and specially define $s^{\text{start}} = 0$ and $s^{\text{goal}} = 2^K - 1$. Furthermore, we use $s[k]$ to refer to the k-th bit of status s, and define $\mathcal{U}_s = \{k = 1, \ldots, K \mid s[k] = 0\}$ as the set of unvisited key areas for status s.

For any possible node–status pair $(n, s) \in \mathcal{N} \times \mathcal{S}$, let $\mathcal{T}_{n,s}$ denote the tentative set of cost vectors at node n and status s. The tentative set $\mathcal{T}_{n,s}$ stores the non-dominated cost vectors of all the identified paths from the start node–status pair $(n^{\text{start}}, s^{\text{start}})$ to the node–status pair (n, s) at any moment during the search. After the MOA* algorithm terminates, the tentative set of the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$ is exactly the Pareto front that we pursue, i.e., $\mathcal{Y}^* = \mathcal{T}_{n^{\text{goal}}, s^{\text{goal}}}$.

To meet the requirements of multimodal optimization, we need to record all the Pareto optimal paths from the start node–status pair $(n^{\text{start}}, s^{\text{start}})$ to any intermediate node–status pair (n, s). To this end, we use $\mathcal{P}_{n,s,c}$ to denote the list of predecessors that lead to the node–status–cost triplet (n, s, c) for any $c \in \mathcal{T}_{n,s}$. Every predecessor in $\mathcal{P}_{n,s,c}$ is of the form (n', s', c', e), where (n', s', c') is the preceding node–status–cost triplet, and e is the edge involved in the transition.

With the above definitions, every node–status–cost triplet (n, s, c) represents a group of paths that all start from the start node n^{start} with status s^{start}, end at node n with status s, and have a cumulative cost c. At each iteration of the main loop, the algorithm needs to determine which node–status–cost triplet is the "best" to expand next. Specifically, the algorithm should select the triplet (n, s, c) that has a non-dominated estimated cost $\hat{f} = c + \textsc{Heuristic}(n, s)$, where HEURISTIC is a heuristic function that estimates the ideal cost (i.e., lower bound of the costs) from the current node–status pair (n, s) to the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$. We will explain the HEURISTIC function later in Section III-C.

The pseudocode of the proposed MOA* algorithm is rendered in Algorithm 2. The algorithm uses a priority queue $\mathcal{Q}$ to maintain all the node–status–cost triplets that need to be expanded. At the beginning, the start cost $c^{\text{start}} = cost[x^{\text{start}}][y^{\text{start}}]$ is inserted to the tentative set $\mathcal{T}_{n^{\text{start}}, s^{\text{start}}}$, and the triplet $(n^{\text{start}}, s^{\text{start}}, c^{\text{start}})$ is pushed to the priority queue $\mathcal{Q}$. At each iteration of the algorithm, the triplet (n, s, c) with the lexicographically smallest estimated cost $\hat{f} = c + \textsc{Heuristic}(n, s)$ among all the triplets in $\mathcal{Q}$ is selected. By doing this, we ensure that the chosen triplet always has a non-dominated estimated cost.

If the chosen triplet (n, s, c) already reaches the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$, it implies that c must be a Pareto optimal point in the objective space. Here, we do not filter out in time those cost vectors in any tentative set that are dominated by the newly identified Pareto optimal point. Instead, we employ a lazy filtering strategy that actually filters out those dominated costs when they are later popped from the priority queue, as was done in [18]. In this case, the current iteration terminates.

If the estimated cost $\hat{f}$ is dominated by (any existing cost vector in) the goal tentative set $\mathcal{T}_{n^{\text{goal}}, s^{\text{goal}}}$, it implies that it is impossible to find any Pareto optimal solutions by expanding the current triplet. According to the lazy filtering strategy, we now remove c from the tentative set $\mathcal{T}_{n,s}$, and delete the predecessor list $\mathcal{P}_{n,s,c}$ accordingly. In this case, the current iteration terminates.

If the above two cases are not involved, the current triplet (n, s, c) is expanded as follows. For each edge $e \in \Delta_n$ incident with the current node n, the succeeding triplet (n', s', c') is determined. There are three possible situations:

- If c' is dominated by $\mathcal{T}_{n',s'}$, it implies that c' is not optimal for the node–status pair (n', s');
- If c' already exists in the tentative set $\mathcal{T}_{n',s'}$, then (n, s, c, e) is added as a new predecessor of the triplet (n', s', c');
- If c' is neither dominated by $\mathcal{T}_{n',s'}$ nor present in this set, we first eliminate all the unpromising costs $\tilde{c}$ from $\mathcal{T}_{n',s'}$ that are dominated by c', delete $\mathcal{P}_{n',s',\tilde{c}}$, and also remove the triplets $(n', s', \tilde{c})$ from $\mathcal{Q}$ if they remain in the queue. Then, the estimated cost $\hat{f}' = c' + \textsc{Heuristic}(n', s')$

is computed. If $\hat{f}'$ is dominated by the goal tentative set $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$, it implies that it is impossible to find any Pareto optimal solutions by expanding the succeeding triplet (n', s', c'), thus it should be ignored. Otherwise, $\mathcal{T}_{n',s'}$, $\mathcal{P}_{n',s',c'}$, and $\mathcal{Q}$ are updated accordingly.

The MOA* algorithm terminates when $\mathcal{Q}$ becomes empty. As aforementioned, the goal tentative set $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ is exactly the Pareto front $\mathcal{Y}^*$ for the MMOPP problem. Algorithm 3 describes a procedure to construct the entire Pareto set $\mathcal{X}^*$ using the backtracking strategy, based on the obtained Pareto front $\mathcal{Y}^*$ and the dictionary of predecessor lists $\mathcal{P}$. Note that the symbol $\oplus$ denotes the concatenation of sequences, and the set Ψ is used to avoid repetitive node–status pairs in the constructed path, thereby preventing from zero-cost cycles.

Algorithm 2. Multi-objective A* algorithm.

global $\mathcal{T}$: Dictionary of tentative sets;
global $\mathcal{P}$: Dictionary of predecessor lists;

$c^{\text{start}} \leftarrow cost[x^{\text{start}}][y^{\text{start}}]$;
Insert c^{start} to $\mathcal{T}_{n^{\text{start}},s^{\text{start}}}$;
Create an empty priority queue $\mathcal{Q}$;
Push $(n^{\text{start}}, s^{\text{start}}, c^{\text{start}})$ to $\mathcal{Q}$;
while $\mathcal{Q} \neq \emptyset$ **do**
  Pop (n, s, c) with the lexicographically smallest $\hat{f} = c +$ HEURISTIC(n, s) from $\mathcal{Q}$;
  if $n = n^{\text{goal}}$ **and** $s = s^{\text{goal}}$ **then**
    continue;
  end if
  if $\hat{f}$ is dominated by $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **then**
    Remove c from $\mathcal{T}_{n,s}$;
    Delete $\mathcal{P}_{n,s,c}$;
    continue;
  end if
  for $e \in \Delta_n$ **do**
    $n' \leftarrow \delta_{n,e}$;
    $s' \leftarrow s$;
    if $\exists k \in \{1, \ldots, K\}$ such that $n' = n_k^{\text{key}}$ **then**
      $s'[k] \leftarrow 1$;
    end if
    $c' \leftarrow c + \omega_e + cost[x_{n'}][y_{n'}]$;
    if c' is dominated by $\mathcal{T}_{n',s'}$ **then**
      continue;
    end if
    if $c' \in \mathcal{T}_{n',s'}$ **then**
      Append (n, s, c, e) to $\mathcal{P}_{n',s',c'}$;
      continue;
    end if
    for $\tilde{c} \in \mathcal{T}_{n',s'}$ that is dominated by c' **do**
      Remove $\tilde{c}$ from $\mathcal{T}_{n',s'}$;
      Delete $\mathcal{P}_{n',s',\tilde{c}}$;
      if $(n', s', \tilde{c}) \in \mathcal{Q}$ **then**
        Remove $(n', s', \tilde{c})$ from $\mathcal{Q}$;
      end if
    end for
    $\hat{f}' \leftarrow c' +$ HEURISTIC(n', s');
    if $\hat{f}'$ is dominated by $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **then**
      continue;
    end if
    Insert c' to $\mathcal{T}_{n',s'}$;
    Append (n, s, c, e) to $\mathcal{P}_{n',s',c'}$;
    Push (n', s', c') to $\mathcal{Q}$;
  end for
end while

Algorithm 3. Construction of the Pareto set.

require $\mathcal{Y}^*$: Pareto front;
require $\mathcal{P}$: Dictionary of predecessor lists;
global $\mathcal{X}^*$: Pareto set, initially empty;

for $c^{\text{goal}} \in \mathcal{Y}^*$ **do**
  $\Pi \leftarrow [(x^{\text{goal}}, y^{\text{goal}})]$;
  $\Psi \leftarrow \{(n^{\text{goal}}, s^{\text{goal}})\}$;
  BUILDPATH$(n^{\text{goal}}, s^{\text{goal}}, c^{\text{goal}}, \Pi, \Psi)$;
end for

procedure BUILDPATH(n, s, c, Π, Ψ) ▷ Recursive
  if $n = n^{\text{start}}$ **and** $s = s^{\text{start}}$ **then**
    Insert Π to $\mathcal{X}^*$;
  else
    for $(n', s', c', e) \in \mathcal{P}_{n,s,c}$ **do**
      if $(n', s') \notin \Psi$ **then**
        $\Pi' \leftarrow [(x_{n'}, y_{n'})] \oplus \Omega_{n',e} \oplus \Pi$;
        $\Psi' \leftarrow \Psi \cup \{(n', s')\}$;
        BUILDPATH$(n', s', c', \Pi', \Psi')$;
      end if
    end for
  end if
end procedure

C. Heuristic Function

In the proposed MOA* algorithm, the HEURISTIC function provides a lower bound estimation of the ideal cost from any node–status pair (n, s) to the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$. A tight lower bound estimation is important not only for guiding the search order, but also for pruning unnecessary branches during the search.

First, a matrix of distance vectors between some nodes is pre-computed. For each dimension $i \in \{1, \ldots, M\}$ and each node $n^{\text{dst}} \in \{n^{\text{goal}}, n_1^{\text{key}}, \ldots, n_K^{\text{key}}\}$, we solve a single-objective shortest path problem on $\mathcal{G}$ to compute the minimum cost between any node $n^{\text{src}} \in \mathcal{N}$ and node n^{dst} considering only the i-th objective function, denoted by $dist[n^{\text{src}}][n^{\text{dst}}][i]$. Note that the distance $dist[n^{\text{src}}][n^{\text{dst}}][i]$ does not include the costs of the end nodes n^{src} and n^{dst}.

Algorithm 4 outlines the HEURISTIC function. The input of the HEURISTIC function is the node–status pair (n, s) whose ideal cost is to be estimated. If (n, s) equals to the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$, the function returns 0. Otherwise, computing the ideal cost in each dimension is equivalent to finding the Hamiltonian path with the least total distance that starts from node n, visits all the key nodes in $\mathcal{U}_s$, and ends at node n^{goal}. However, due to the exponential time complexity of the particular shortest Hamiltonian path problem, it is impractical to find the exactly least total distance when $\mathcal{U}_s$ is large.

To make a trade-off between the estimation strength and computational efficiency, we propose a hybrid strategy as follows. The ideal cost h is first initialized by accumulating the cost of the goal node and the costs of all the unvisited key nodes. Then, the ideal cost h is further strengthened by adding the minimum total distance from the current node n to

Algorithm 4. Heuristic function.

require *dist*: Matrix of minimum costs between nodes;

function HEURISTIC(n, s)
  if $n = n^{\text{goal}}$ **and** $s = s^{\text{goal}}$ **then**
    return 0;
  end if
  $h \leftarrow cost[x^{\text{goal}}][y^{\text{goal}}]$;
  for $k \in \mathcal{U}_s$ **do**
    $h \leftarrow h + cost[x_k^{\text{key}}][y_k^{\text{key}}]$;
  end for
  if $|\mathcal{U}_s| = 0$ **then**
    $h \leftarrow h + dist[n][n^{\text{goal}}]$;
  else if $|\mathcal{U}_s| = 1$ **then**
    Let u be the only key node in $\mathcal{U}_s$;
    $h \leftarrow h + dist[n][u] + dist[u][n^{\text{goal}}]$;
  else if $|\mathcal{U}_s| = 2$ **then**
    Let u and v be the two key nodes in $\mathcal{U}_s$;
    for $i = 1, \ldots, M$ **do**
      $\sigma_1 \leftarrow dist[n][u][i] + dist[v][n^{\text{goal}}][i]$;
      $\sigma_2 \leftarrow dist[n][v][i] + dist[u][n^{\text{goal}}][i]$;
      $h[i] \leftarrow h[i] + \min(\sigma_1, \sigma_2) + dist[u][v][i]$;
    end for
  else
    for $i = 1, \ldots, M$ **do**
      $\alpha \leftarrow \min_{v \in \mathcal{U}_s} dist[n][v][i]$;
      $\beta \leftarrow \min_{v \in \mathcal{U}_s} dist[v][n^{\text{goal}}][i]$;
      $h[i] \leftarrow h[i] + \alpha + \beta + \text{MINSPANTREE}(i, \mathcal{U}_s)$;
    end for
  end if
  return h;
end function

the goal node n^{goal} while passing through all these unvisited key nodes.

- If $|\mathcal{U}_s| = 0$, the minimum total distance is simply given by $dist[n][n^{\text{goal}}]$;
- If $|\mathcal{U}_s| = 1$, the minimum total distance is given by $dist[n][u] + dist[u][n^{\text{goal}}]$, where u is the only node in $\mathcal{U}_s$;
- If $|\mathcal{U}_s| = 2$, let u and v denote the two key nodes in $\mathcal{U}_s$. Then, we compare the only two possible permutations $n \to u \to v \to n^{\text{goal}}$ and $n \to v \to u \to n^{\text{goal}}$, and choose the one with less distance for each dimension $i \in \{1, \ldots, M\}$;
- If $|\mathcal{U}_s| \geq 3$, for each dimension $i \in \{1, \ldots, M\}$, we solve a minimum spanning tree (MST) problem to compute a lower bound on the minimum total distance for connecting all the unvisited key nodes in $\mathcal{U}_s$. Then, the i-th dimension of the ideal cost h is increased by the minimum distance from node n to the MST, the minimum distance from the MST to node n^{goal}, and the total distance within the MST.

Next, we present a brief proof on the admissibility of the proposed MOA* algorithm. The above algorithm description already implies that the heuristic function is admissible, that is, it never overestimates the actual cost from a given node–status pair to the goal node–status pair. For each Pareto optimal solution path, any of its prefix sub-paths will not be dominated by other paths in the same tentative set. In addition, due to the admissibility of the HEURISTIC function, the estimated cost of any prefix sub-path will not be dominated by existing Pareto points in $\mathcal{T}_{n^{\text{goal}}, s^{\text{goal}}}$ either. Therefore, all Pareto optimal solution paths will be obtained when the proposed MOA* algorithm terminates.

IV. COMPUTATIONAL RESULTS

All the algorithms are implemented in C++ and compiled on Ubuntu 20.04 under the Windows Subsystem for Linux. In our program, the tentative sets $\mathcal{T}_{n,s}$ for $(n, s) \in \mathcal{N} \times \mathcal{S}$ are implemented using the `pareto::front` container from a third-party library named `pareto`, which provides $O(M \log |\mathcal{T}_{n,s}|)$ insert, delete, lookup, and dominance check operations.[1] The priority queue $\mathcal{Q}$ is implemented using the `std::map` container with a lexicographical comparator, which provides $O(M \log |\mathcal{Q}|)$ insert, delete, and lookup operations. The hardware environment is a desktop computer running Windows 10 with an Intel Core i5-8500 CPU and 8 gigabytes of RAM.

Table I presents the computational results for the official MMOPP test suite [19]. For each test problem, the column "Original Map" gives the number of passable areas and the number of adjacent pairs among the passable areas in the original map. Similarly, the column "Reduced Map" gives the number of retained areas and the number of adjacent pairs among the retained areas in the reduced map. In the column "Graph Model", the size of the graph model is presented in terms of the number of nodes and the number of edges. The next columns "MOA* Iterations", "Runtime (in ms)", and "Complexity Indicator" demonstrate the computational efficiency of the proposed approach. The number of MOA* iterations is equivalent to the number of triplets that have been pushed in or popped from the priority queue. The runtime refers to the total time spent in the entire computation from the start of the feasibility check to the completion of the Pareto set. The complexity indicator is computed by dividing the runtime by the reference runtime, and the reference runtime is measured by running the test program described in Fig. 5. The last two columns give the cardinalities of the Pareto front and the Pareto set finally obtained.

```
double x = 0.55;
for (int i = 1; i <= 1000000; i++) {
    x = x + x;
    x = x / 2;
    x = x * x;
    x = sqrt(x);
    x = log(x);
    x = exp(x);
    x = x / (x + 2);
}
```

Fig. 5. Test program for measuring the reference runtime.

The results show that the graph modeling technique significantly reduces the scale of the problem by around 10 times. Based on the graph model, the proposed MOA* algorithm exactly solves almost all the test problems in few milliseconds,

[1] The `pareto` library is available at https://alandefreitas.github.io/pareto/.

TABLE I
COMPUTATIONAL RESULTS FOR THE TEST PROBLEMS

Test Problem	Parameters (M, K)	Original Map (#areas, #adjs)	Reduced Map (#areas, #adjs)	Graph Model ($\|\mathcal{N}\|$, $\|\mathcal{E}\|$)	MOA* Iterations	Runtime (in ms)	Complexity Indicator	Pareto Front $\|\mathcal{Y}^*\|$	Pareto Set $\|\mathcal{X}^*\|$
1	(2, 0)	(380, 400)	(380, 400)	(35, 55)	55	0.33	0.031	4	9
2	(3, 0)	(377, 405)	(377, 405)	(37, 65)	59	0.31	0.029	7	24
3	(3, 0)	(623, 669)	(612, 658)	(57, 103)	61	0.32	0.030	4	13
4	(3, 0)	(616, 652)	(566, 603)	(49, 86)	75	0.28	0.026	7	9
5	(3, 0)	(1727, 1825)	(1689, 1789)	(118, 218)	192	0.85	0.080	5	24
6	(2, 0)	(380, 400)	(380, 400)	(35, 55)	38	0.22	0.020	3	5
7	(3, 0)	(377, 405)	(377, 405)	(37, 65)	102	0.32	0.030	12	16
8	(4, 0)	(623, 669)	(612, 658)	(57, 103)	296	1.15	0.107	36	48
9	(5, 0)	(616, 652)	(566, 603)	(49, 86)	445	2.04	0.190	81	105
10	(7, 0)	(1727, 1825)	(1689, 1789)	(118, 218)	6482	56.37	5.260	1070	1280
11	(2, 1)	(380, 400)	(380, 400)	(35, 55)	26	0.18	0.017	2	4
12	(3, 2)	(377, 405)	(377, 405)	(37, 65)	160	0.50	0.047	10	22

TABLE II
COMPARISON OF THE NAIVE, BLIND, AND PROPOSED APPROACHES

Test Suite		Naive Approach		Blind Approach		Proposed Approach	
Test Problem	Parameters (M, K)	MOA* Iterations	Runtime (in ms)	MOA* Iterations	Runtime (in ms)	MOA* Iterations	Runtime (in ms)
1	(2, 0)	588	1.01	65	0.34	55	0.33
2	(3, 0)	699	1.03	72	0.32	59	0.31
3	(3, 0)	1203	1.70	110	0.38	61	0.32
4	(3, 0)	1391	1.63	113	0.35	75	0.28
5	(3, 0)	4067	6.51	285	0.96	192	0.85
6	(2, 0)	638	0.62	62	0.21	38	0.22
7	(3, 0)	1830	2.61	179	0.51	102	0.32
8	(4, 0)	4679	10.49	546	1.88	296	1.15
9	(5, 0)	7280	23.02	761	3.20	445	2.04
10	(7, 0)	133548	779.41	12833	94.64	6482	56.37
11	(2, 1)	948	0.89	91	0.26	26	0.18
12	(3, 2)	8879	12.72	922	2.07	160	0.50

with a complexity indicator smaller than 0.2. The only exception is test problem 10, which involves 7 objective functions and consists of 118 nodes and 218 edges in the graph model. It takes 56.37 milliseconds (with a complexity indicator of 5.26) and 6482 iterations in the MOA* algorithm.

To further examine the impacts of the graph modeling technique and the heuristic function on the efficiency of the MOA* algorithm, we additionally compare the proposed approach to another two degenerated versions called "naive approach" and "blind approach". The heuristic function is not used in both versions, so they are in fact uninformed search. In the naive approach, the undirected graph is modeled directly based on the original grid map, while the blind approach uses the same graph modeling technique as done by the proposed approach.

Table II compares the efficiency of the three approaches, all of which are able to exactly solve the test problems. Compared with the naive approach, the blind approach takes about one tenth of the number of MOA* iterations, which is consistent with the scale reduction of the problem by the graph modeling technique. By comparing the blind approach and the proposed approach, we can find that the heuristic function has nearly doubled the search efficiency on large-scale problems.

V. CONCLUSIONS

In this paper, we propose an efficient approach to exactly solve the MMOPP problems. The proposed approach first models the original grid map into a concise undirected graph. Then, it applies a well-designed MOA* algorithm to find all the Pareto optimal solution paths on the obtained graph model. Computational experiments show that the proposed approach is considerably efficient in exactly solving the 12

test problems in the official MMOPP test suite. We also show the admissibility of the proposed approach so that its computational results can be used as the standard answers to the test suite.

The proposed approach is not limited to the original problem configuration specified in [19]. In the original configuration, costs are only assigned to passable areas; there can also be corresponding costs for the movements between adjacent passable areas. The grid shape of the original map is not necessary, as the MOA* algorithm is applicable to any undirected graph. In addition, it is not difficult to convert the proposed approach to asymmetric MMOPP problems with appropriate modifications, in which paths may not exist in both directions or the cumulative costs might be different.

There is a possible avenue of future research for further improving the efficiency of the MOA* algorithm. Recall that in the MOA* algorithm, if the succeeding triplet (n', s', c') is pruned if c' is dominated by $\mathcal{T}_{n',s'}$. This pruning condition can be enhanced by checking whether c' is dominated by any tentative set $\mathcal{T}_{n',s^*}$ such that $\mathcal{U}_{s^*} \subseteq \mathcal{U}_{s'}$. However, such enhancement requires extra dominance check effort, which should also be taken into account.

REFERENCES

[1] K. Deb, "Multi-objective optimisation using evolutionary algorithms: An introduction," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, L. Wang, A. H. C. Ng, and K. Deb, Eds. London: Springer, 2011, pp. 3–34.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-Report*, vol. 103, 2001.

[4] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[5] A. Zhou, Q. Zhang, and Y. Jin, "Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1167–1189, 2009.

[6] M. Preuss, C. Kausch, C. Bouvy, and F. Henrich, "Decision space diversity can be essential for solving multiobjective real-world problems," in *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, Eds. Berlin, Heidelberg: Springer, 2010, pp. 367–377.

[7] R. Tanabe and H. Ishibuchi, "A review of evolutionary multimodal multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 193–200, 2020.

[8] P. P.-Y. Wu, D. Campbell, and T. Merz, "Multi-objective four-dimensional vehicle motion planning in large dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 621–634, 2011.

[9] P. Dell'Olmo, M. Gentili, and A. Scozzari, "On finding dissimilar Pareto-optimal paths," *European Journal of Operational Research*, vol. 162, no. 1, pp. 70–82, 2005.

[10] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, no. 2, pp. 293–309, 2008.

[11] P. Hansen, "Bicriterion path problems," in *Multiple Criteria Decision Making Theory and Application*, G. Fandel and T. Gal, Eds. Berlin, Heidelberg: Springer, 1980, pp. 109–127.

[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[13] E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, 1984.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[15] B. S. Stewart and C. C. White, III, "Multiobjective A*," *Journal of the ACM*, vol. 38, no. 4, pp. 775–814, 1991.

[16] L. Mandow and J. L. Pérez de la Cruz, "Multiobjective A* search with consistent heuristics," *Journal of the ACM*, vol. 57, no. 5, pp. 1–25, 2010.

[17] E. Machuca, L. Mandow, J. L. Pérez de la Cruz, and A. Ruiz-Sepulveda, "A comparison of heuristic best-first algorithms for bicriterion shortest path problems," *European Journal of Operational Research*, vol. 217, no. 1, pp. 44–53, 2012.

[18] 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.

[19] J. Liang, C. Yue, G. Li, B. Qu, P. N. Suganthan, and K. Yu, "Problem definitions and evaluation criteria for the CEC 2021 on multimodal multiobjective path planning optimization," 2020.

[20] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.