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

A presentation at IEEE CEC 2021

Bo Jin

College of Management, Shenzhen University, Shenzhen, China

jinbo@szu.edu.cn

July 1, 2021

Table of contents

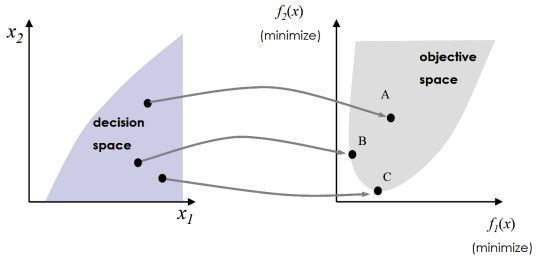
Background

Multi-objective optimization

A multi-objective optimization problem deals with more than one objective function:

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

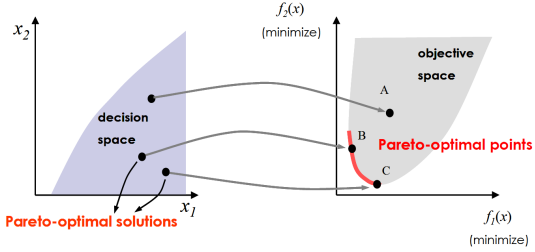
- Objective functions $\boldsymbol{f} = (f_1, \ldots, f_M)$
- Decision space $\mathcal{X} = \{\boldsymbol{x} \mid \boldsymbol{x} \text{ is feasible}\}$
- Objective space $\mathcal{Y} = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \text{ is feasible}\}$



Pareto optimality

Pareto optimality

A feasible solution $\boldsymbol{x} \in \mathcal{X}$ (resp., objective point $\boldsymbol{y} \in \mathcal{Y}$) is said to be *Pareto optimal* if it is not dominated by any other feasible solutions in $\mathcal{X}$ (resp., objective points in $\mathcal{Y}$).



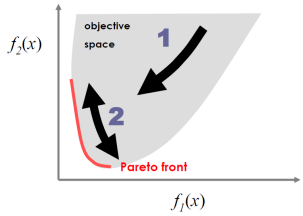
Pareto set and Pareto front

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 $\mathcal{Y}^*$.

Goals in multi-objective optimization

Two goals in multi-objective optimization:

1. To find a set of solutions as *close* as possible to the Pareto front. — *This goal is mandatory.*
2. To find a set of solutions as *diverse* as possible. — *Diversity can be defined in both decision and objective spaces.*

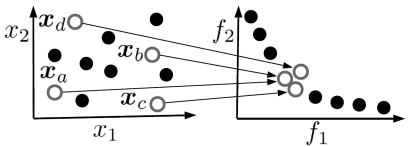


Popular multi-objective evolutionary algorithms:

- NSGA-II — *non-dominated sorting genetic algorithm II*
- SPEA2 — *strength Pareto evolutionary algorithm 2*
- MOEA/D — *multi-objective evolutionary algorithm based on decomposition*
- ...

Multimodal multi-objective optimization

However, most existing multi-objective evolutionary algorithms only aim at approximating the Pareto front.



Four solutions are identical or close to each other in the objective space but are far from each other in the solution space.

Approximating only the Pareto front is not enough!

Multimodal multi-objective optimization

A multimodal multi-objective optimization problem involves finding all solutions that are equivalent to Pareto optimal solutions. — *"Multimodal" means that multiple solutions correspond to one Pareto optimal point.*

MMOPP Optimization

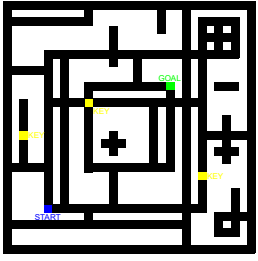
MMOPP optimization

深圳大学 SHENZHEN UNIVERSITY

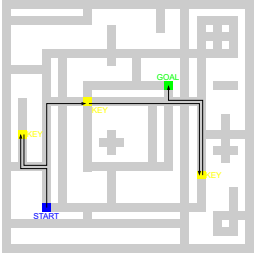
2021 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION 28.06–1.07.2021 • Kraków • POLAND

The *multimodal multi-objective path planning* (MMOPP) optimization is discussed in SS-37 & C-8 in IEEE CEC 2021.

The MMOPP optimization aims at finding *all the Pareto optimal paths* from *a start area* to *a goal area* on a grid map with respect to *multiple objectives*, while these paths are required to pass through *several must-visit areas* in an arbitrary order.



Example map



Example path

MMOPP test suite (Liang et al., 2020)

Relation to the shortest path problem

Clearly, MMOPP is a special extension of the shortest path problem with the following attributes:

1. one source node,
2. one destination node, — *informed search can be used*
3. costs assigned to nodes, — *easy to handle*
4. multiple objectives, — *large search space!*
5. several must-visit nodes. — *larger search space!!*

Notations for MMOPP

Notation	Description
(x, y)	area located at the x-th column from the left and the y-th row from the top
$map[x][y]$	0 means that area (x, y) is passable, and 1 means the opposite
M	number of objectives
$cost[x][y]$	M-dimensional non-negative cost vector associated with area (x, y)
$(x^{\text{start}}, y^{\text{start}})$	start area
$(x^{\text{goal}}, y^{\text{goal}})$	goal area
K	number of key areas
$(x_k^{\text{key}}, y_k^{\text{key}})$	k-th key area (the index k does not represent its visit order)

*Must-visit areas are hereinafter called key areas.

*The start area, the goal area, and the key areas are collectively referred to as the mandatory areas.

Fundamental Concepts

Appetizer

Before formally introducing the proposed approach, we first review some existing studies on the *multi-objective shortest path problem*, and outline the general form of the *multi-objective A** (MOA*) algorithm.

[*] Non-negative edge costs are assumed. For algorithms that deal with negative edge costs, please refer to Bellman–Ford algorithm or other label-correcting algorithms. The overall process is nevertheless similar.

Existing approaches

One-to-all shortest path problem:

- Dijkstra's algorithm (Dijkstra, 1959) — *single objective*
- Hansen (1980) — *two objectives*
- Martins (1984) — *multiple objectives*

One-to-one shortest path problem:

- A* algorithm (Hart et al., 1968) — *single objective*
- Stewart & White (1991) — *first attempt of MOA* based on node expansion*
- NAMOA* (Mandow & Pérez de la Cruz, 2010) — *new approach to MOA* based on label expansion*
- $\text{NAMOA}^*_{\text{dr}}$ (Pulido et al., 2015) — *improved NAMOA* with a faster dominance check*

General A* algorithm

Symbol	Description
$\mathcal{N}$	set of nodes
n^{start}	start node (i.e., source node)
n^{goal}	goal node (i.e., destination node)
Δ_n	edges incident with node n
$\delta_{n,e}$	end of edge e incident with node n
ω_e	cost of edge e
h	heuristic function • admissible ⇒ correct algorithm • consistent ⇒ fewest iterations

Variable	Description
$\mathcal{Q}$	open set (i.e., priority queue)
τ_n	tentative cost of node n
$h(n)$	estimated cost from n to n^{goal}
$\mathcal{P}_n$	predecessor list of node n

```
1  τ_n ← ∞ for all n ∈ N;
2  Q ← ∅;
3  τ_{n^start} ← 0;
4  Push n^start to Q;
5  while Q ≠ ∅ do
6      Pop n with min τ_n + h(n) from Q;
7      if n = n^goal then
8          break;
9      for e ∈ Δ_n do
10         n' ← δ_{n,e};
11         c' ← τ_n + ω_e;
12         if c' < τ_{n'} then
13             τ_{n'} ← c';
14             P_{n'} ← {n};
15             Push n' to Q if n' ∉ Q;
16         else if c' = τ_{n'} then
17             Append n to P_{n'};
```

*We temporarily assume no parallel edges or node costs.

Extending A* to MOA* (1/2)

- ω_e is now the cost vector of edge e.
- ~~t_n~~ $\mathcal{T}_n$: tentative set of non-dominated costs of node n.
- ~~p_n~~ $\mathcal{P}_{n,c}$: predecessor list of node n with cost $c \in \mathcal{T}_n$.
- $\mathcal{Q}$ is now a set of node–cost pairs to be expanded.

Extending A* to MOA* (2/2)

- The algorithm terminates only when $\mathcal{Q}$ becomes empty.
- At each iteration, a node–cost pair (n, c) with a non-dominated estimated cost $c + h(n)$ is popped from $\mathcal{Q}$.
- To simplify the implementation, we choose the one with the *lexicographically smallest* estimated cost.
- A node–cost pair (n, c) is *filtered* if its estimated cost $c + h(n)$ is dominated by (any cost in) $\mathcal{T}_{n^{\text{goal}}}$, or simply $c + h(n) \succ \mathcal{T}_{n^{\text{goal}}}$.
- A node–cost pair (n, c) is *pruned* if its current cost c is dominated by (any cost in) $\mathcal{T}_n$, or simply $c \succ \mathcal{T}_n$.
- After the algorithm terminates, the finally obtained $\mathcal{T}_{n^{\text{goal}}}$ is exactly the Pareto front $\mathcal{Y}^*$.

*Parallel edges now make sense.

**We also assume no node costs.

General MOA* algorithm

1 $\mathcal{T}_n \leftarrow \emptyset$ for all $n \in \mathcal{N}$;
2 $\mathcal{Q} \leftarrow \emptyset$;
3 Insert $\mathbf{0}$ to $\mathcal{T}_{n^{\text{start}}}$;
4 Push $(n^{\text{start}}, \mathbf{0})$ to $\mathcal{Q}$;
5 **while** $\mathcal{Q} \neq \emptyset$ **do**
6 Pop (n, c) with lex-min $c + h(n)$ from $\mathcal{Q}$;
7 **if** $n = n^{\text{goal}}$ **then**
8 **continue**;
9 **if** $c + h(n) \succ \mathcal{T}_{n^{\text{goal}}}$ **then** // *filter* (n, c)
10 Remove c from $\mathcal{T}_n$;
11 Delete $\mathcal{P}_{n,c}$;
12 **continue**;
13–27 {label expansion}

13 **for** $e \in \Delta_n$ **do** {label expansion}
14 $n' \leftarrow \delta_{n,e}$;
15 $c' \leftarrow c + \omega_e$;
16 **if** $c' \succ \mathcal{T}_{n'}$ **then** // *prune* (n', c')
17 **continue**;
18 **if** $c' \in \mathcal{T}_{n'}$ **then**
19 Append (n, c, e) to $\mathcal{P}_{n',c'}$;
20 **continue**;
21 **for** $\tilde{c} \in \mathcal{T}_{n'} : \tilde{c} \succ c'$ **do** // *prune* $(n', \tilde{c})$
22 Remove $\tilde{c}$ from $\mathcal{T}_{n'}$;
23 Delete $\mathcal{P}_{n',c'}$;
24 Remove $(n', \tilde{c})$ from $\mathcal{Q}$;
25 **if** $c' + h(n') \succ \mathcal{T}_{n^{\text{goal}}}$ **then** // *filter* (n', c')
26 **continue**;
27 Insert c' to $\mathcal{T}_{n'}$;
28 Append (n, c, e) to $\mathcal{P}_{n',c'}$;
29 Push (n', c') to $\mathcal{Q}$;

Properties of MOA*

Admissibility

An admissible heuristic function h ensures the admissibility of MOA*.

Consistency

A consistent heuristic function h ensures that a closed (expanded) label will never be open (pushed to $\mathcal{Q}$) again.

Efficiency

Greater values of $h(n)$ can push the evaluation of more nodes beyond the real optimal cost, hence reducing search effort.

* A comprehensive theoretical discussion is given by Mandow & Pérez de la Cruz (2010).

What's next?

- Formulate MMOPP into a graph model;
- Extend MOA* to handle node costs and must-visit nodes;
- Propose an admissible heuristic function (better if it is consistent);
- Write an efficient program.

Proposed Approach

Outline

1. Feasibility check — *check whether all mandatory areas are reachable*
2. Map reduction — *remove unnecessary areas to reduce the map*
3. Graph modeling — *further reduce the scale of the problem*
4. Informed search — *compute the Pareto front* $\mathcal{Y}^*$
5. Path reconstruction — *construct the Pareto set* $\mathcal{X}^*$

Map reduction (1/3)

The map can be reduced by removing unnecessary areas that will never appear in a Pareto optimal solution path:

- *Unreachable areas* are unnecessary.
- A connected group of *non-mandatory areas* separated by an *articulation point* are unnecessary.

Articulation point

An articulation point (or cut vertex) is any node in a graph whose removal increases the number of connected components.

Map reduction (2/3)

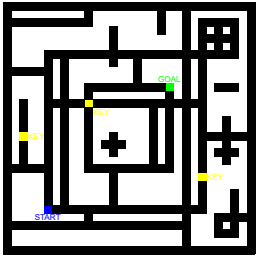
Starting from the start area, build a depth-first search (DFS) spanning tree of reachable areas, and compute the following variables:

- $depth[x][y]$: depth of area (x, y) in the DFS spanning tree;
- $low[x][y]$: depth of the farthest ancestor that is adjacent to any area in the subtree rooted at area (x, y);
- $flag[x][y]$: boolean value that indicates whether the subtree rooted at area (x, y) contains any mandatory areas.

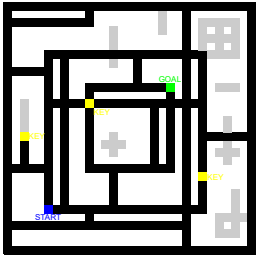
Map reduction (3/3)

Consider an area (x, y) in the DFS spanning tree:

- If any of its child node (x', y') satisfies $low[x'][y'] \geq depth[x][y]$ and $flag[x'][y'] = \texttt{false}$, then the entire subtree rooted at (x', y') can be pruned.



Original map



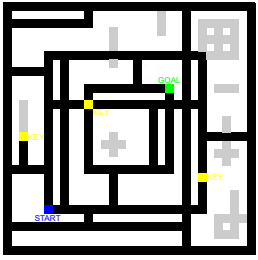
Reduced map

*Tarjan's algorithm for finding articulation points (Tarjan, 1972).

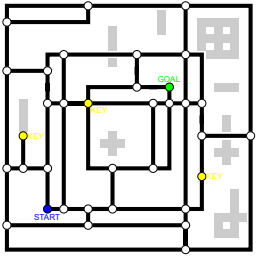
Graph modeling

Based on the reduced map, we model the MMOPP into an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ as follows.

- Each node $n \in \mathcal{N}$ corresponds to *a mandatory area* or *a retained area of degree 3 or 4*.
- Each edge $e \in \mathcal{E}$ corresponds to *a sequence of continuous retained areas of degree 2 between two nodes*.



Reduced map



Graph model

Extra notations for MMOPP

深圳大學 SHENZHEN UNIVERSITY

Notation	Description
$\mathcal{N}$	set of nodes
n^{start}	start node
n^{goal}	goal node
n_k^{key}	the k-th key node
(x_n, y_n)	coordinate of node n
Δ_n	set of edges incident with node n
$\delta_{n,e}$	the other end of edge e incident with node e
ω_e	cumulative cost of all the intermediate areas in edge e
$\Omega_{n,e}$	sequence of intermediate areas from node n to node $\delta_{n,e}$

Adapting MOA* to MMOPP (1/2)

- A status indicates which key areas have been visited along a path.
- $\mathcal{S} = \{0, \ldots, 2^K - 1\}$ is the set of all the 2^K possible statuses.
- Specially, define $s^{\text{start}} = 0$ and $s^{\text{goal}} = 2^K - 1$.
- $\mathcal{U}_s = \{k = 1, \ldots, K \mid s[k] = 0\}$ denotes the set of unvisited key areas for status s.
- ~~$\mathcal{T}_n$~~ $\mathcal{T}_{n,s}$: non-dominated set of tentative costs at node n with status s.
- ~~$\mathcal{P}_{n,c}$~~ $\mathcal{P}_{n,s,c}$: predecessor list of node n with status s and cost $c \in \mathcal{T}_{n,s}$.
- $\mathcal{Q}$ is now a set of node–status–cost triplets to be expanded.
- Do not forget node costs.

Adapting MOA* to MMOPP (2/2)

- The algorithm terminates only when $\mathcal{Q}$ becomes empty.
- At each iteration, the node–status–cost triplet (n, s, c) with the *lexicographically smallest* estimated cost $c + h(n, s)$ is popped from $\mathcal{Q}$.
- A node–status–cost triplet (n, s, c) is *filtered* if its estimated cost $c + h(n, s)$ is dominated by (any cost in) $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$, or simply $c + h(n) \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$.
- A node–status–cost triplet (n, s, c) is *pruned* if its current cost c is dominated by (any cost in) $\mathcal{T}_{n,s}$, or simply $c \succ \mathcal{T}_{n,s}$.
- After the algorithm terminates, the finally obtained $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ is exactly the Pareto front $\mathcal{Y}^*$.

MOA* for MMOPP

1 $\mathcal{T}_{n,s} \leftarrow \emptyset$ for all $(n,s) \in \mathcal{N} \times \mathcal{S}$;
2 $\mathcal{Q} \leftarrow \emptyset$;
3 $c^{\text{start}} \leftarrow cost[x^{\text{start}}][y^{\text{start}}]$;
4 Insert c^{start} to $\mathcal{T}_{n^{\text{start}},s^{\text{start}}}$;
5 Push $(n^{\text{start}}, s^{\text{start}}, c^{\text{start}})$ to $\mathcal{Q}$;
6 **while** $\mathcal{Q} \neq \emptyset$ **do**
7 Pop (n,s,c) with lex-min $c + h(n,s)$ from $\mathcal{Q}$;
8 **if** $n = n^{\text{goal}}$ **and** $s = s^{\text{goal}}$ **then**
9 **continue**;
10 **if** $c + h(n,s) \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **then**
 // filter (n,s,c)
11 Remove c from $\mathcal{T}_{n,s}$;
12 Delete $\mathcal{P}_{n,s,c}$;
13 **continue**;
14–33 {label expansion}

14 **for** $e \in \Delta_n$ **do** {label expansion}
15 $n' \leftarrow \delta_{n,e}$;
16 $s' \leftarrow s$;
17 **if** $\exists k \in \{1,\dots,K\} : n' = n_k^{\text{key}}$ **then**
18 $s'[k] \leftarrow 1$;
19 $c' \leftarrow c + \omega_e + cost[x_{n'}][y_{n'}]$;
20 **if** $c' \succ \mathcal{T}_{n',s'}$ **then** *// prune* (n',s',c')
21 **continue**;
22 **if** $c' \in \mathcal{T}_{n',s'}$ **then**
23 Append (n,s,c,e) to $\mathcal{P}_{n',s',c'}$;
24 **continue**;
25 **for** $\tilde{c} \in \mathcal{T}_{n',s'} : \tilde{c} \succ c'$ **do** *// prune* $(n',s',\tilde{c})$
26 Remove $\tilde{c}$ from $\mathcal{T}_{n',s'}$;
27 Delete $\mathcal{P}_{n',s',\tilde{c}}$;
28 Remove $(n',s',\tilde{c})$ from $\mathcal{Q}$;
29 **if** $c' + h(n',s') \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **then** *// filter* (n',s',c')
30 **continue**;
31 Insert c' to $\mathcal{T}_{n',s'}$;
32 Append (n,s,c,e) to $\mathcal{P}_{n',s',c'}$;
33 Push (n',s',c') to $\mathcal{Q}$;

Data structures

The program is written in C++.

Variable	Role	Data structure	Time complexity
$\mathcal{Q}$	open set (priority queue)	`std::map` (red–black tree)	$O(M \log \lvert\mathcal{Q}\rvert)$ insert, delete, lookup
$\mathcal{T}_{n,s}$	tentative set (non-dominated set)	`pareto::front` (R-tree)	$O(M \log \lvert\mathcal{T}_{n,s}\rvert)$ insert, delete, lookup, dominance check

[*] We use `std::map` to implement the open set $\mathcal{Q}$ as a priority queue with a lexicographical comparator.

[**] The `pareto` library is available online at https://alandefreitas.github.io/pareto/ (by Alan de Freitas).

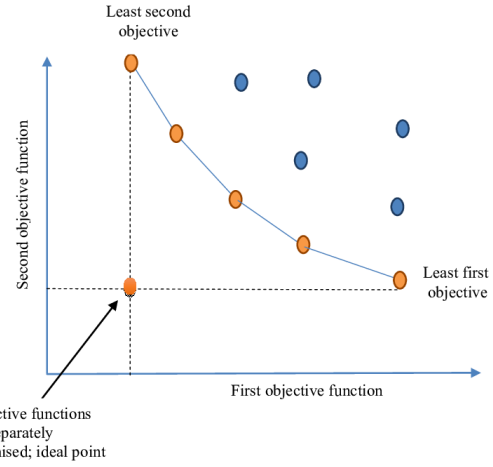
Path reconstruction

$\mathcal{X}^* \leftarrow \emptyset$;
for $c^{\text{goal}} \in \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **do**
  $\Pi \leftarrow [(x^{\text{goal}}, y^{\text{goal}})]$;
  $\Psi \leftarrow \{(n^{\text{goal}}, s^{\text{goal}})\}$;
  BuildPath$(n^{\text{goal}}, s^{\text{goal}}, c^{\text{goal}}, \Pi, \Psi)$;
function 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$(n', s', c', \Pi', \Psi')$;

The operator $\oplus$ 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.

Heuristic function (1/4)

The heuristic function $h(n, s)$ gives a *lower bound* estimation of the *ideal cost* from node–status pair (n, s) to the goal node–status pair $(n^{\text{goal}}, s^{\text{goal}})$.



Heuristic function (2/4)

First, a matrix of distance vectors between some nodes is pre-computed.

- For each dimension $i \in \{1, \dots, M\}$ and each node $n^{\text{dst}} \in \{n^{\text{goal}}, n_1^{\text{key}}, \dots, n_K^{\text{key}}\}$, 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}.

Heuristic function (3/4)

1 **function** $h(n, s)$
2 **if** $n = n^{\text{goal}}$ **and** $s = s^{\text{goal}}$ **then**
3 **return** 0;
4 $\hat{h} \leftarrow cost[x^{\text{goal}}][y^{\text{goal}}]$;
5 **for** $k \in \mathcal{U}_s$ **do**
6 $\hat{h} \leftarrow \hat{h} + cost[x_k^{\text{key}}][y_k^{\text{key}}]$;
7–22 {add estimated edge cost}
23 **return** $\hat{h}$

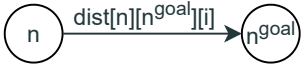
{add estimated edge cost}

7 **if** $|\mathcal{U}_s| = 0$ **then**
8 $\hat{h} \leftarrow \hat{h} + dist[n][n^{\text{goal}}]$;
9 **else if** $|\mathcal{U}_s| = 1$ **then**
10 Let u be the only key node in $\mathcal{U}_s$;
11 $\hat{h} \leftarrow \hat{h} + dist[n][u] + dist[u][n^{\text{goal}}]$;
12 **else if** $|\mathcal{U}_s| = 2$ **then**
13 Let u and v be the two key nodes in $\mathcal{U}_s$;
14 **for** $i = 1, \ldots, M$ **do**
15 $\sigma_1 \leftarrow dist[n][u][i] + dist[v][n^{\text{goal}}][i]$;
16 $\sigma_2 \leftarrow dist[n][v][i] + dist[u][n^{\text{goal}}][i]$;
17 $\hat{h}[i] \leftarrow \hat{h}[i] + \min(\sigma_1, \sigma_2) + dist[u][v][i]$;
18 **else**
19 **for** $i = 1, \ldots, M$ **do**
20 $\alpha \leftarrow \min_{v \in \mathcal{U}_s} dist[n][v][i]$;
21 $\beta \leftarrow \min_{v \in \mathcal{U}_s} dist[v][n^{\text{goal}}][i]$;
22 $\hat{h}[i] \leftarrow \hat{h}[i] + \alpha + \beta + \textsc{MinSpanTree}(i, \mathcal{U}_s)$;

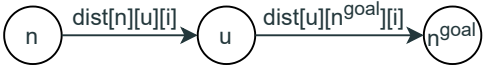
Heuristic function (4/4)

Compute a lower bound of the total edge cost from n to n^{goal} through all the key nodes in $\mathcal{U}_s$, only considering the i-th objective function.

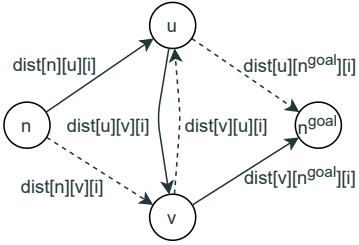
$|\mathcal{U}_s| = 0$:



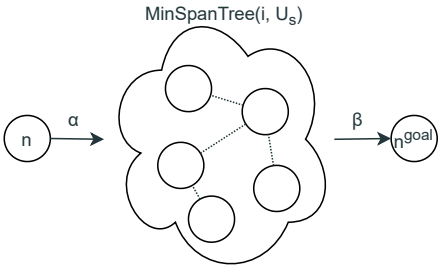
$|\mathcal{U}_s| = 1$:



$|\mathcal{U}_s| = 2$:



$|\mathcal{U}_s| \geq 3$:



Computational results

深圳大学 SHENZHEN UNIVERSITY

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

The hardware environment is a desktop computer running Windows 10 with an Intel Core i5-8500 CPU and 8 GB of RAM.

Impacts of graph modeling and heuristic function

深圳大學 SHENZHEN UNIVERSITY

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

Key factors to high efficiency

- Map reduction and graph modeling — *one tenth of the original scale*
- Efficient filtering and pruning — *sophisticated data structures*
- Strong lower bound estimation — *strengthen filtering*
- Avoid duplicate computation — *trade space for time*
- Programming skills — *third-party libraries*

Conclusion

Future research

Stronger pruning condition for a new triplet:

- Recall that in the MOA* algorithm, 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 $\mathcal{U}_{s^*} \subseteq \mathcal{U}_{s'}$.

Near-optimal solutions are also desirable:

- Two different solutions $\boldsymbol{x}$ and $\boldsymbol{x'}$ are said to be δ-equivalent if $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x'})\| \leq \delta$, where $\|\cdot\|$ is a distance measurement and δ is a non-negative threshold value given by the decision maker.
- The relaxed multimodal multi-objective optimization aims at finding all δ-equivalent Pareto optimal solutions.

Thank you!

Appendix

Timely filtering

1 $\mathcal{T}_{n,s} \leftarrow \emptyset$ for all $(n, s) \in \mathcal{N} \times \mathcal{S}$;
2 $\mathcal{Q} \leftarrow \emptyset$;
3 $c^{\text{start}} \leftarrow cost[x^{\text{start}}][y^{\text{start}}]$;
4 Insert c^{start} to $\mathcal{T}_{n^{\text{start}},s^{\text{start}}}$;
5 Push $(n^{\text{start}}, s^{\text{start}}, c^{\text{start}})$ to $\mathcal{Q}$;
6 **while** $\mathcal{Q} \neq \emptyset$ **do**
7 Pop (n, s, c) with lex-min $c + h(n, s)$ from $\mathcal{Q}$;
8 **if** $n = n^{\text{goal}}$ and $s = s^{\text{goal}}$ **then**
9 **for** $(\tilde{n}, \tilde{s}, \tilde{c}) \in \mathcal{Q} : \tilde{c} + h(\tilde{n}, \tilde{s}) \succ c$ **do**
// filter $(\tilde{n}, \tilde{s}, \tilde{c})$
10 Remove $\tilde{c}$ from $\mathcal{T}_{\tilde{n},\tilde{s}}$;
11 Delete $\mathcal{P}_{\tilde{n},\tilde{s},\tilde{c}}$;
12 Remove $(\tilde{n}, \tilde{s}, \tilde{c})$ from $\mathcal{Q}$;
13 **continue**;
14–33 {label expansion}

Timely filtering

At any moment in the algorithm, there does not exist any $(n, s) \in \mathcal{N} \times \mathcal{S}$ and $c \in \mathcal{T}_{n,s}$ such that $c + h(n, s) \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$.

However, the orange block (lines 8–11) will be relatively slow, unless the open set $\mathcal{Q}$ provides efficient dominance check operations. Another way to do timely filtering is to filter out node–status–cost triplets $(\tilde{n}, \tilde{s}, \tilde{c})$ such that $(\tilde{n}, \tilde{s}) \in \mathcal{N} \times \mathcal{S}$, $\tilde{c} \in \mathcal{T}_{\tilde{n},\tilde{s}}$, and $\tilde{c} + h(\tilde{n}, \tilde{s}) \succ c$, which require extra efforts.

Lazy filtering

1 $\mathcal{T}_{n,s} \leftarrow \emptyset$ for all $(n, s) \in \mathcal{N} \times \mathcal{S}$;
2 $\mathcal{Q} \leftarrow \emptyset$;
3 $c^{\text{start}} \leftarrow cost[x^{\text{start}}][y^{\text{start}}]$;
4 Insert c^{start} to $\mathcal{T}_{n^{\text{start}},s^{\text{start}}}$;
5 Push $(n^{\text{start}}, s^{\text{start}}, c^{\text{start}})$ to $\mathcal{Q}$;
6 **while** $\mathcal{Q} \neq \emptyset$ **do**
7 Pop (n, s, c) with lex-min $c + h(n, s)$ from $\mathcal{Q}$;
8 **if** $n = n^{\text{goal}}$ **and** $s = s^{\text{goal}}$ **then**
9 **continue**;
10 **if** $c + h(n, s) \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ **then**
 // filter (n, s, c)
11 Remove c from $\mathcal{T}_{n,s}$;
12 Delete $\mathcal{P}_{n,s,c}$;
13 **continue**;
14–33 {label expansion}

The green block (lines 9–12) is relatively fast, since it is possible to implement the tentative set $\mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$ with efficient dominance check operations.

Lazy filtering

At any moment in the algorithm, there may exist some $(n, s) \in \mathcal{N} \times \mathcal{S}$ and $c \in \mathcal{T}_{n,s}$ such that $c + h(n, s) \succ \mathcal{T}_{n^{\text{goal}},s^{\text{goal}}}$, but they will always be filtered out eventually.

More discussion on lazy filtering can be found in Sanders & Mandow (2013).

References

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.

Hansen, P. (1980). Bicriterion path problems. In G. Fandel & T. Gal (Eds.), *Multiple Criteria Decision Making Theory and Application* (pp. 109–127). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

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

Machuca, E., Mandow, L., Pérez de la Cruz, J. L., & Ruiz-Sepulveda, A. (2012). A comparison of heuristic best-first algorithms for bicriterion shortest path problems. *European Journal of Operational Research*, 217(1), 44–53.

Mandow, L. & Pérez de la Cruz, J. L. (2010). Multiobjective A* search with consistent heuristics. *Journal of the ACM*, 57(5), 1–25.

Martins, E. Q. V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2), 236–245.

Pulido, F.-J., Mandow, L., & Pérez-de-la-Cruz, J.-L. (2015). Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64, 60–70.

Sanders, P. & Mandow, L. (2013). Parallel label-setting multi-objective shortest path search. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing* (pp. 215–224).

Stewart, B. S. & White, III, C. C. (1991). Multiobjective A*. *Journal of the ACM*, 38(4), 775–814.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2), 146–160.