A Simple and Fast Bi-Objective Search Algorithm

Carlos Hernández[1], William Yeoh[2], Jorge A. Baier[3], Han Zhang[4], Luis Suazo[1] and Sven Koenig[4]

[1]Universidad Andrés Bello, Chile

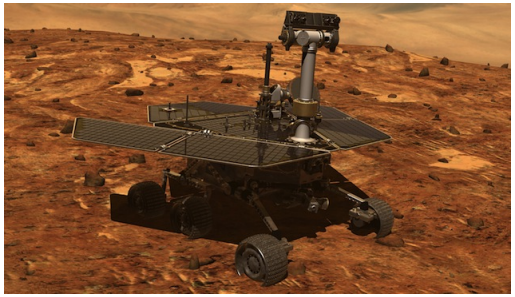
[2]Washington University in St. Louis

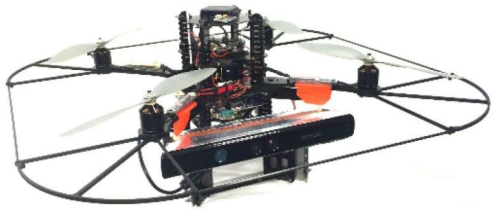
[3]Pontificia Universidad Católica de Chile

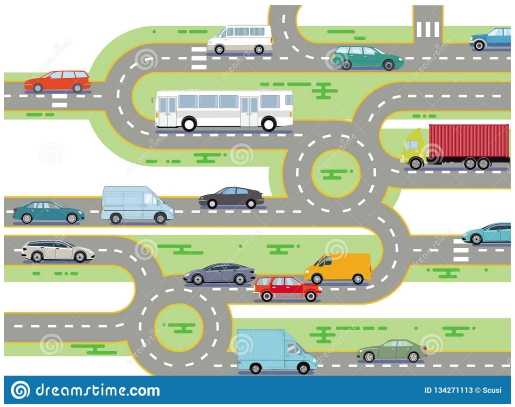
[4]University of Southern California

ICAPS-2020

Bi-objective search has many applications







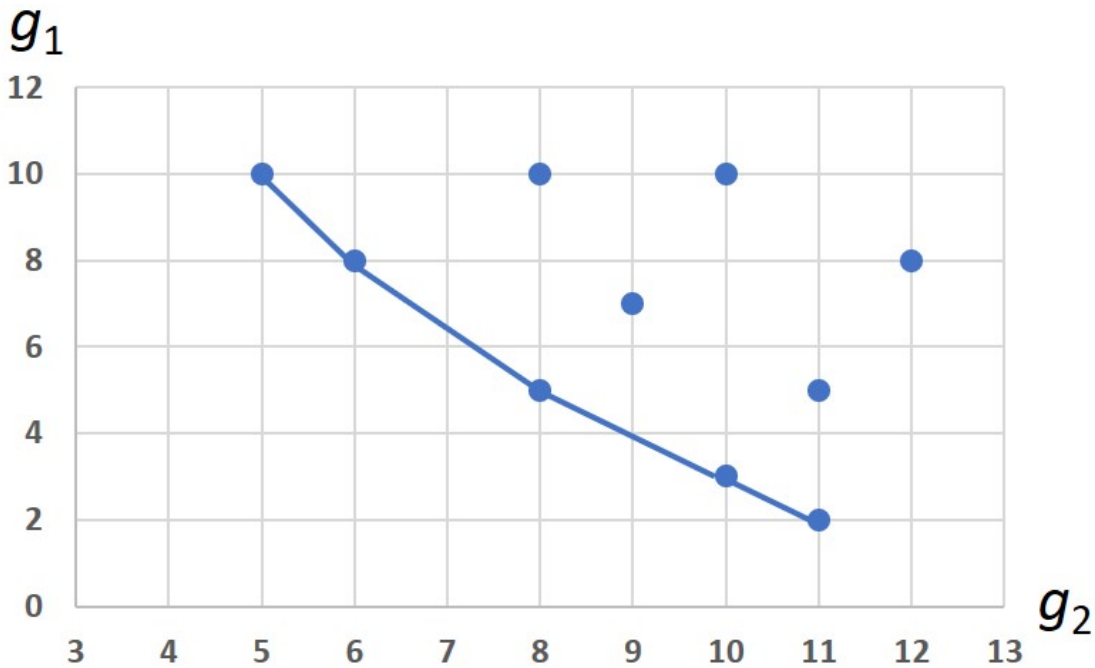
- **Path-planning in robotics**: distance and battery consumption
- **HAZMAT transport in cities**: travel time and risk of exposure for residents
- **Cycling** (path-finding): distance and driver safety
- **Vehicle routing**: monetary cost and travel time

To our knowledge, bi-objective search not supported in PDDL

Bi-objective search

- **Two** objective cost functions g_1, g_2
- *Dominance relation*:

  $(a, b) \prec (a', b')$ iff $a \leq a'$ and $b \leq b'$ but $(a, b) \neq (a', b')$
- Pareto-optimal set: contains all *non-dominated* solutions

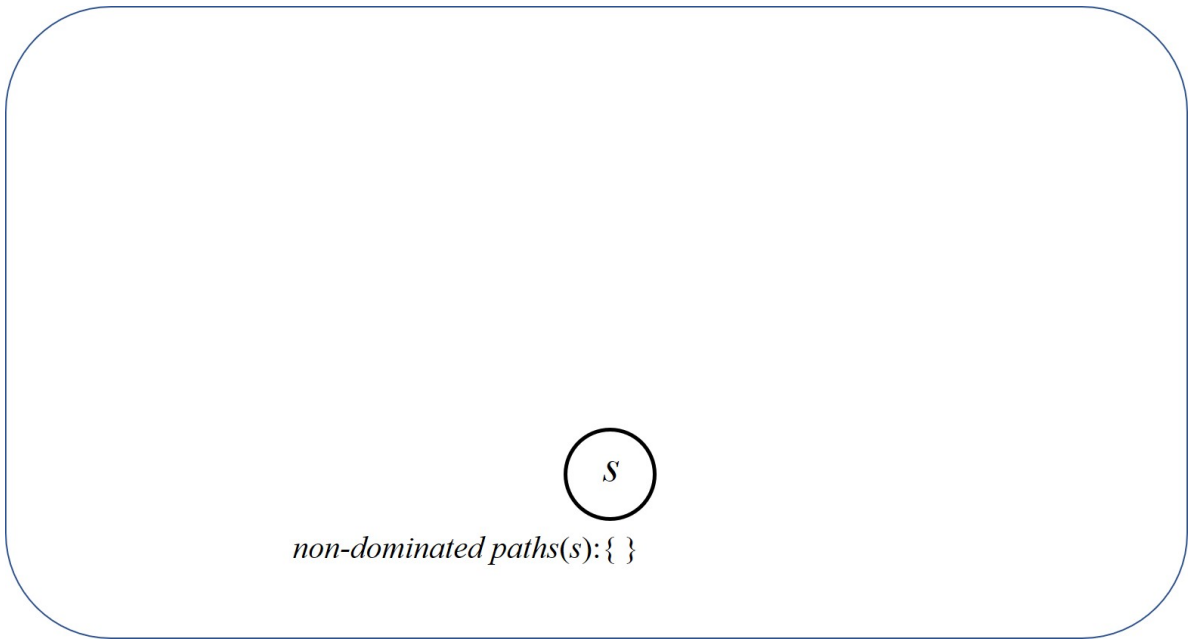


$(10, 5) \prec (10, 8)$, $(10, 5) \not\prec (8, 6)$

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

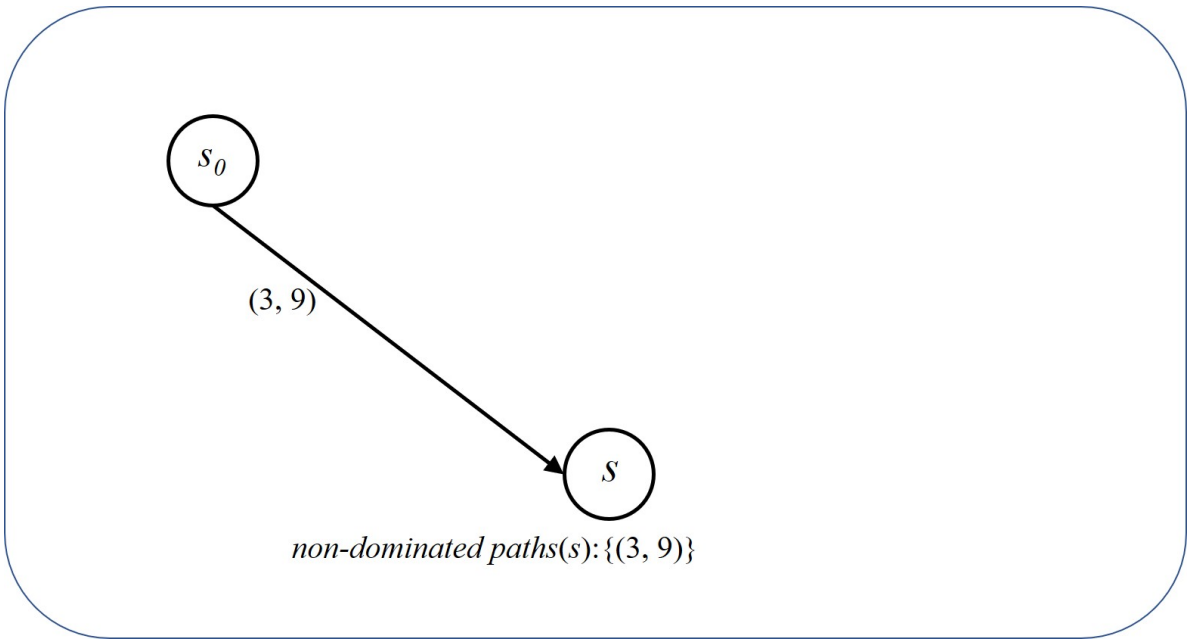


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

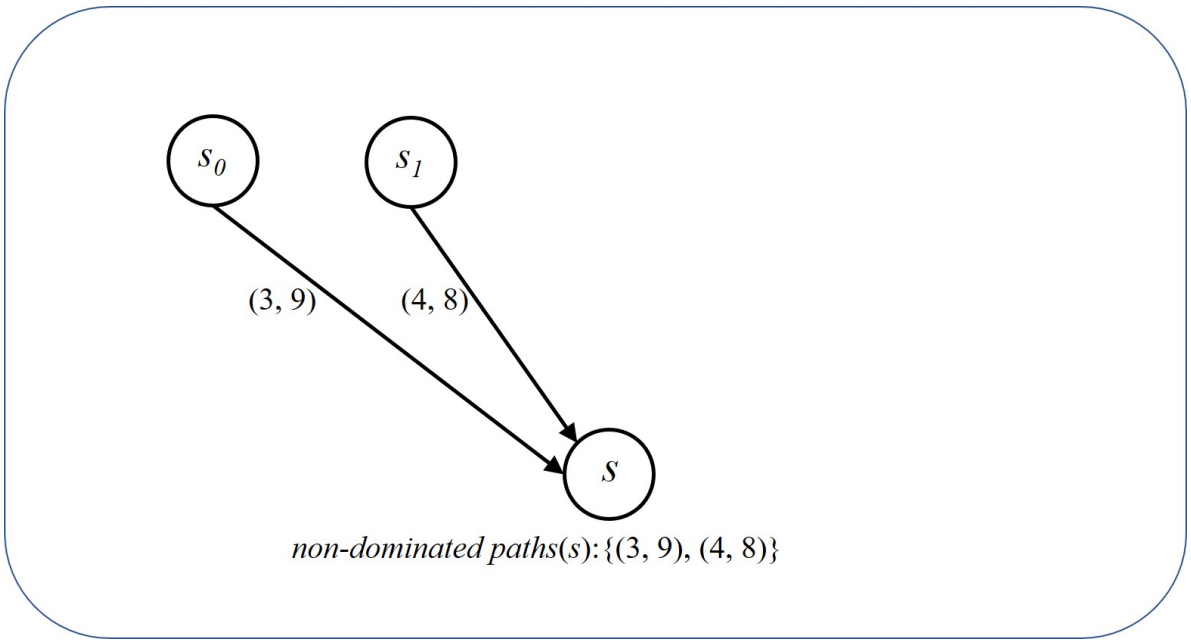


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

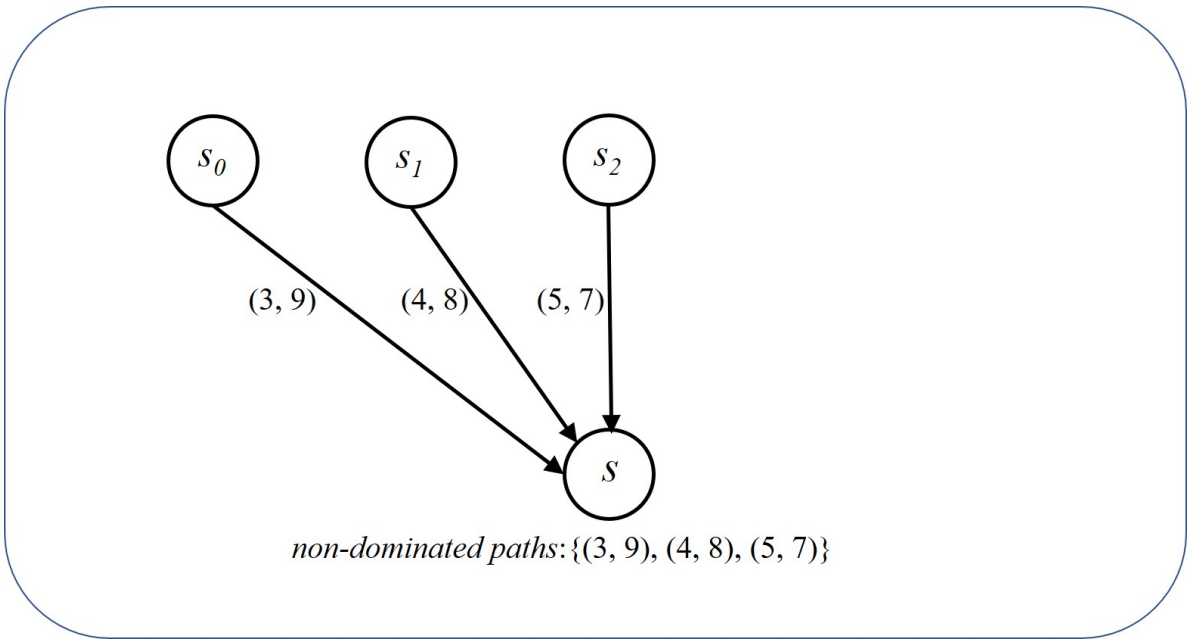


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

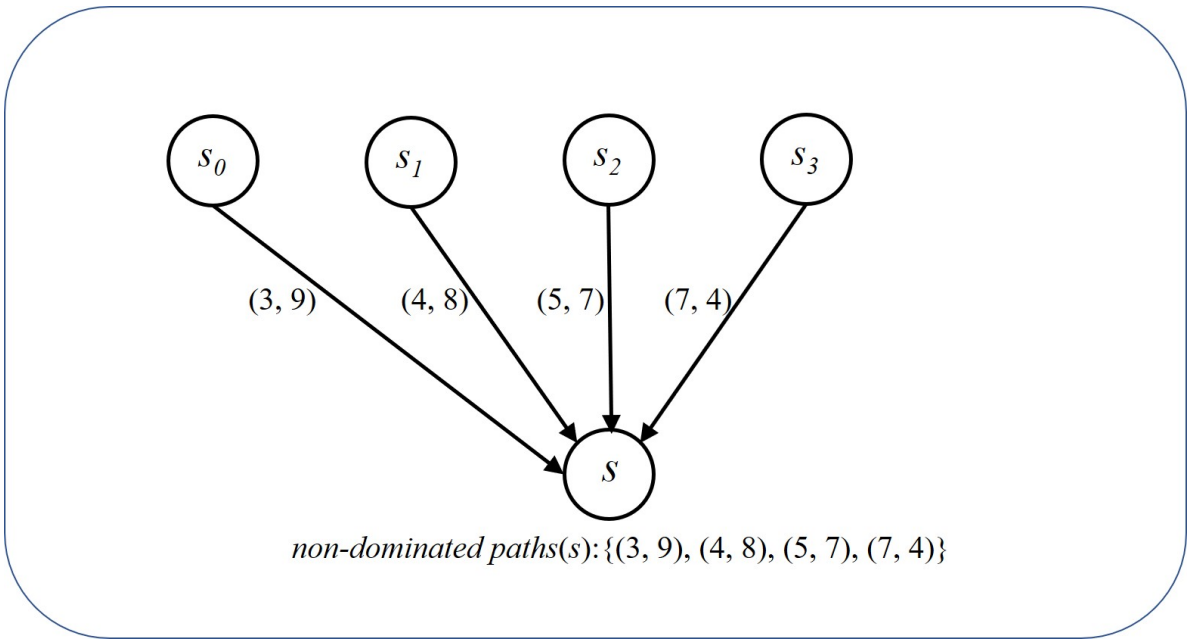


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

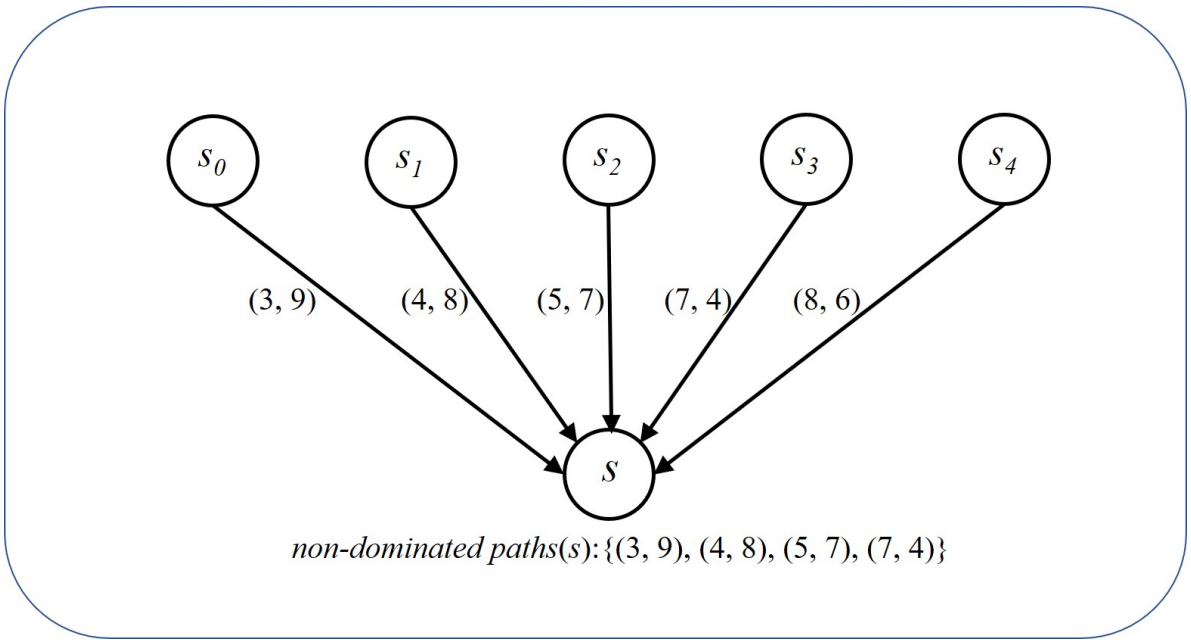


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.

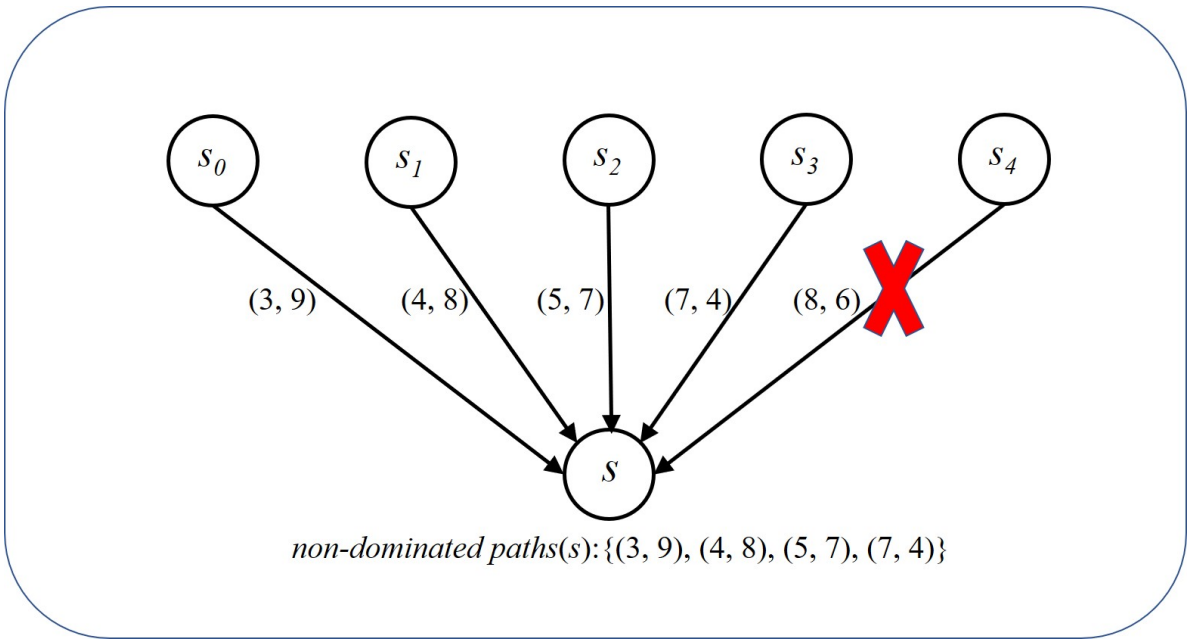


Example

Bi-Objective Search Algorithms

- State-of-the-art NAMOA*dr (Pulido et al., 2015).
- *Dominance check*: Does the newly found path to a state s is *dominate* (or *is dominated by*) a previously found path to s.

  Dominance checking has a big impact in performance.



The process takes linear time.

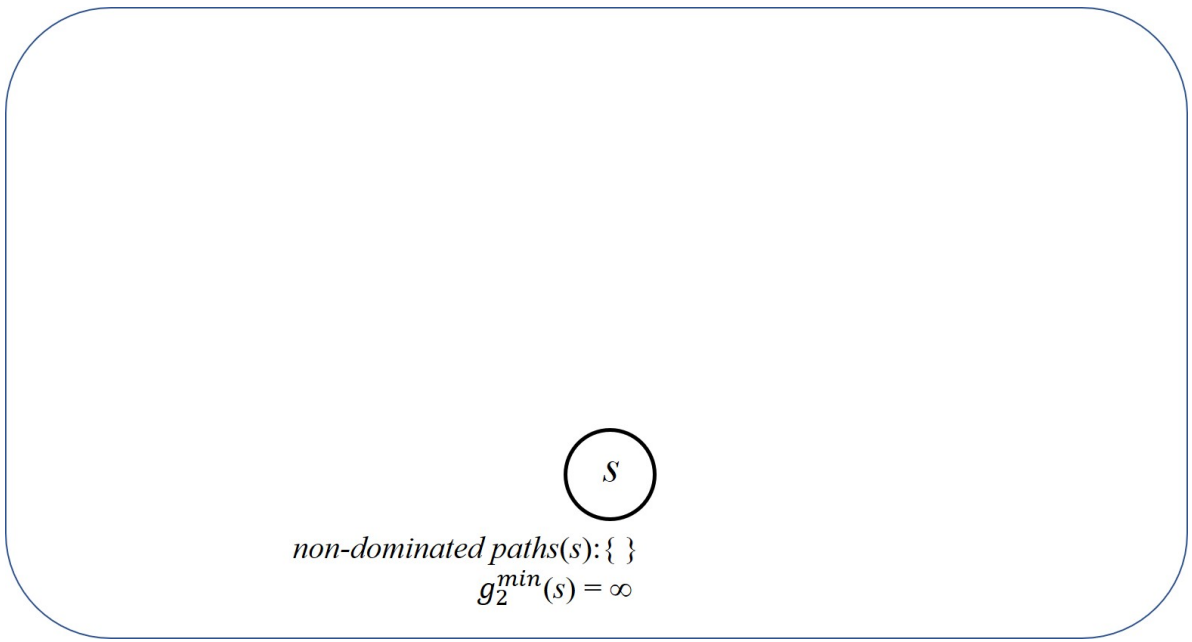
Contribution: a New Bi-Objective Search Algorithm

Highlights of Bi-Objective A* (BOA*)

1. Dominance checking in constant time (instead of linear time).
2. Simple. Resembling standard A*.
3. Empirical evaluation: find a Pareto set orders of magnitude faster.

Bi-Objective A* (Dominance checking)

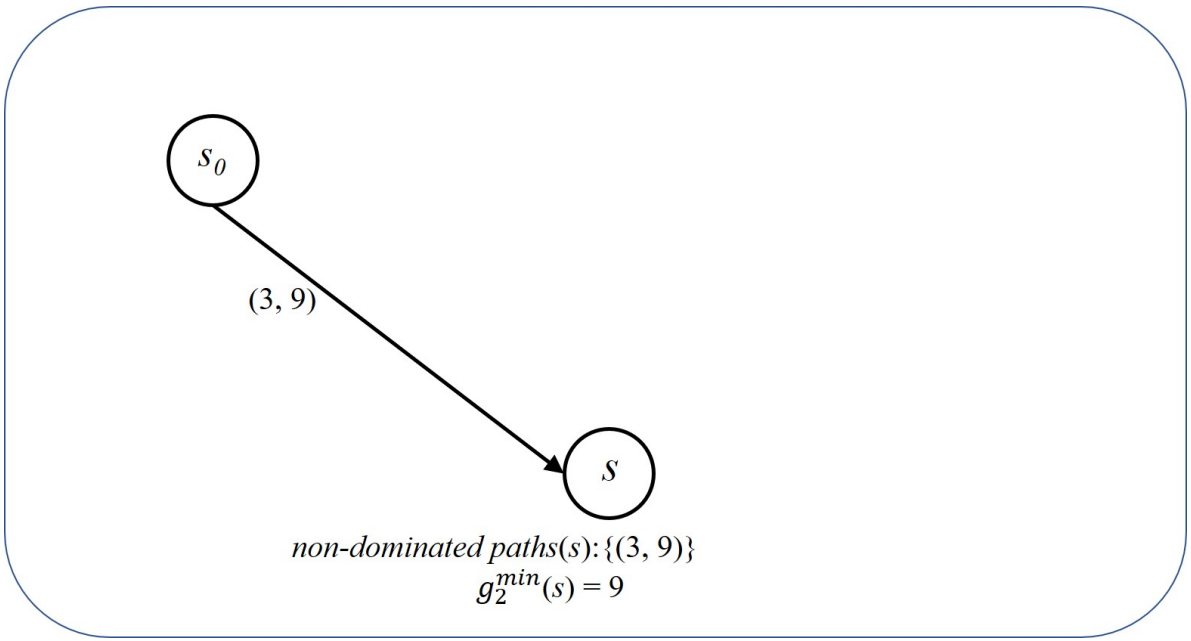
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

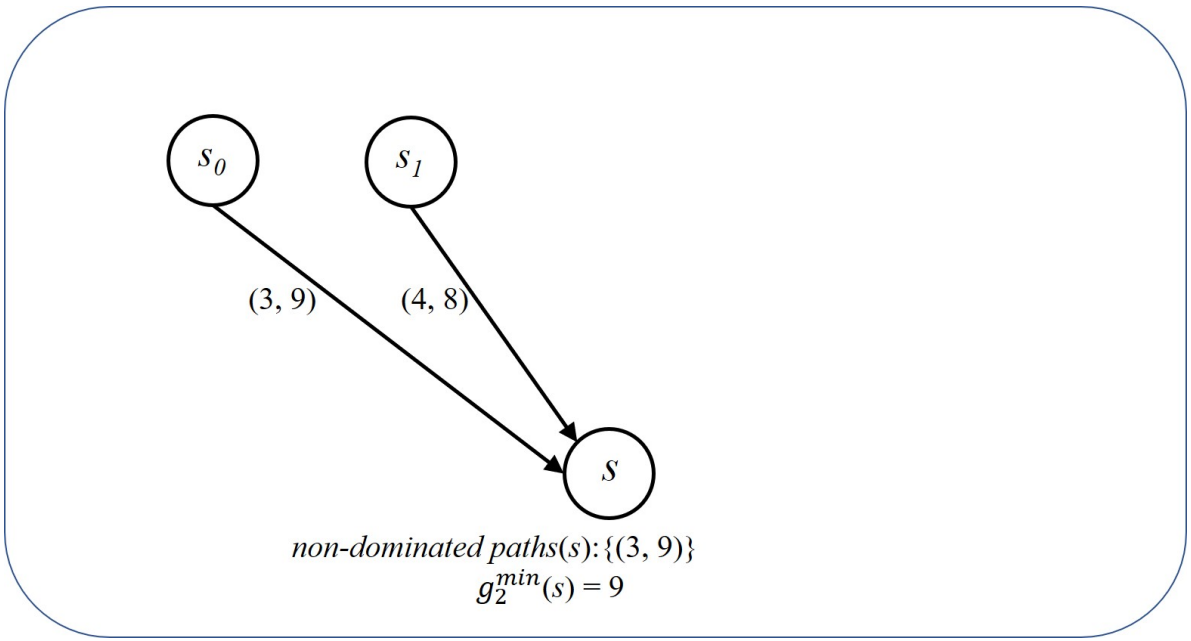
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

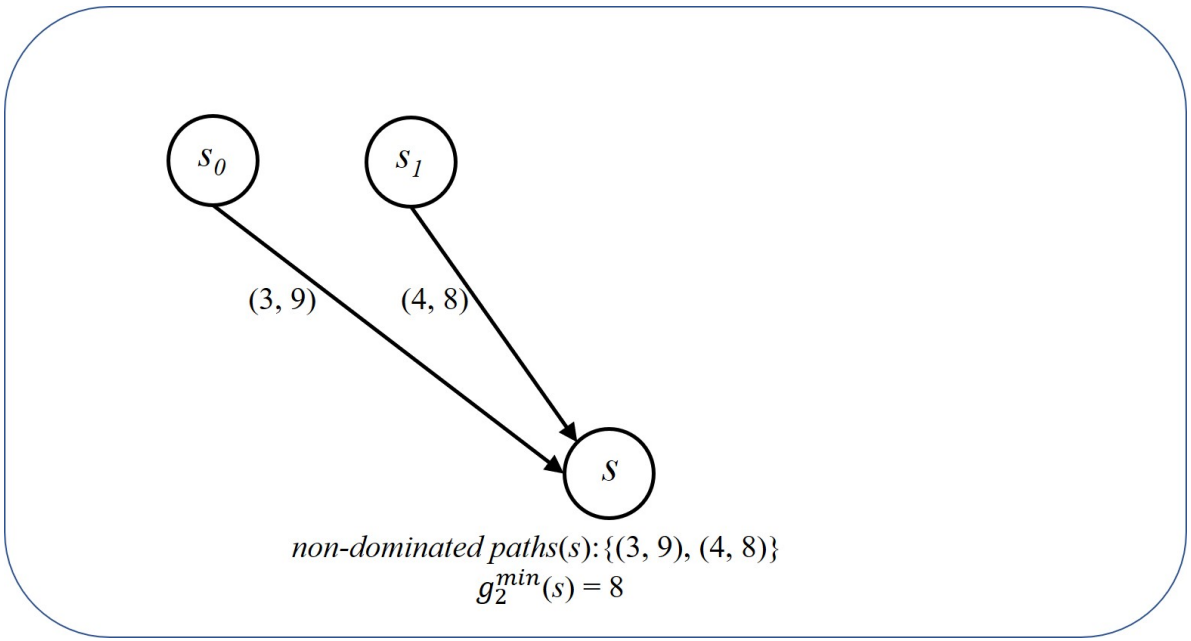
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

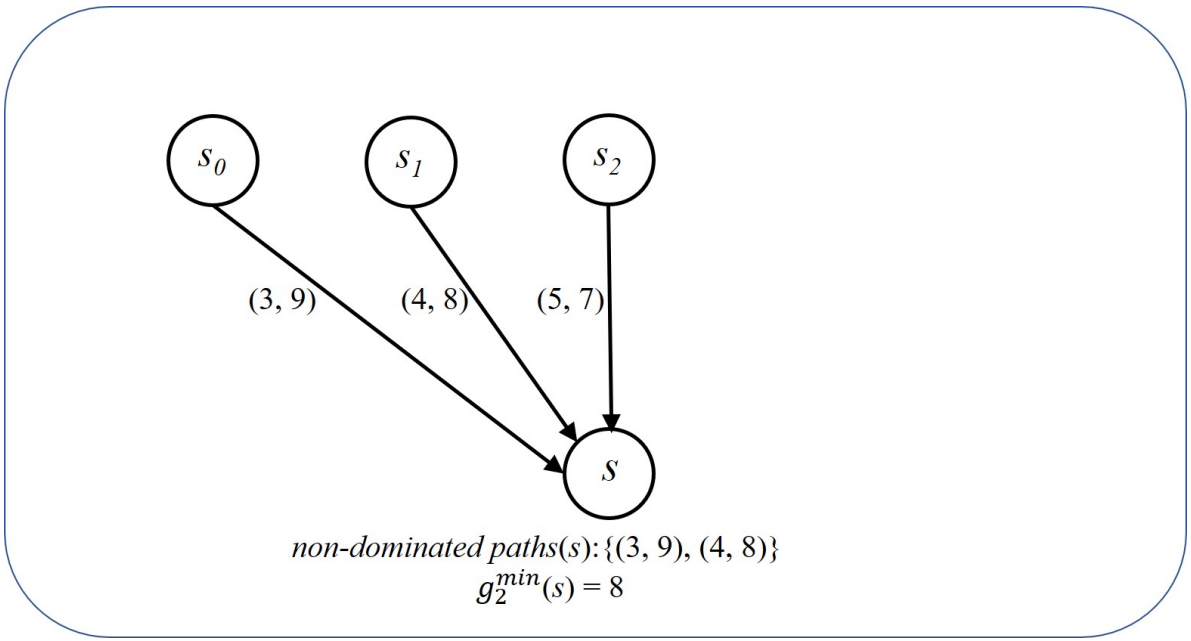
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

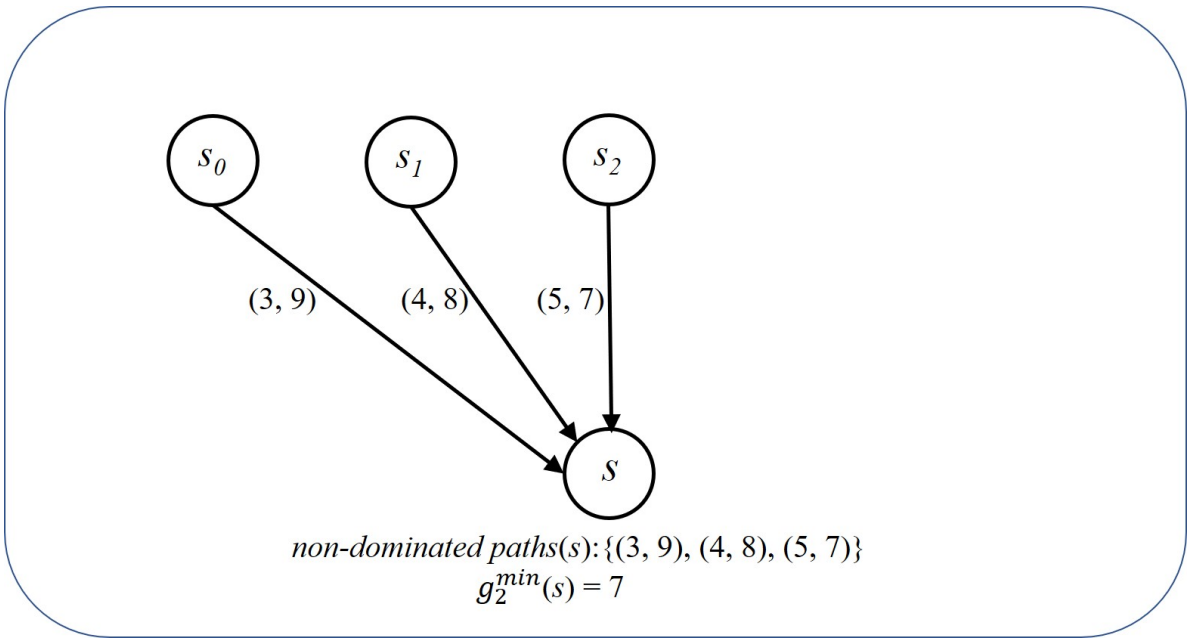
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

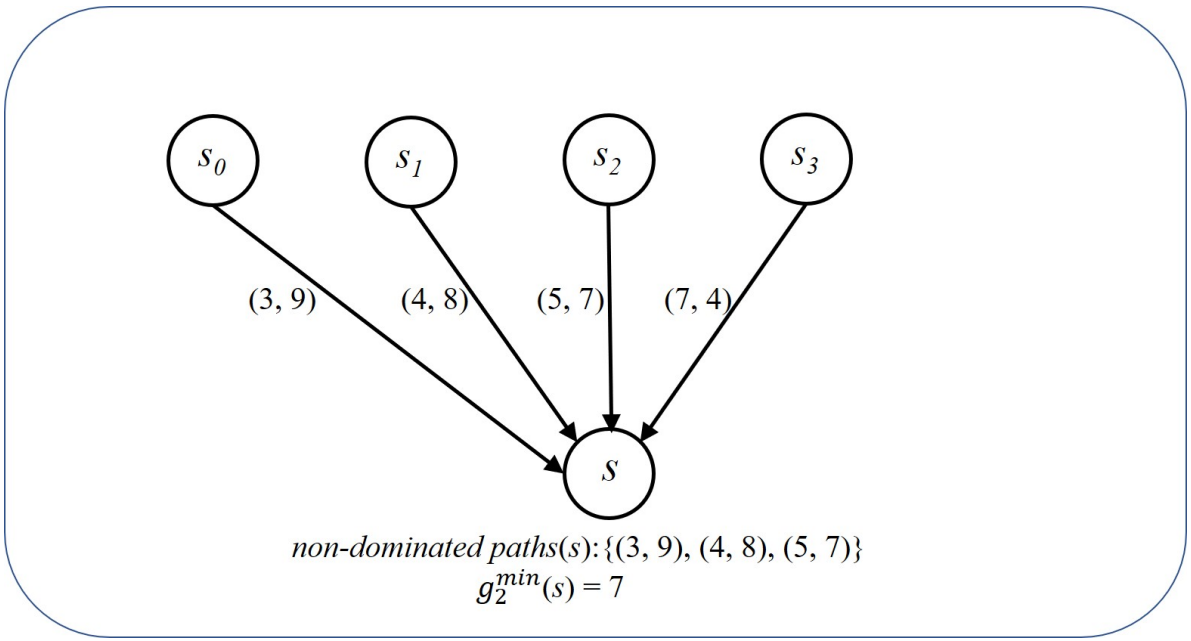
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

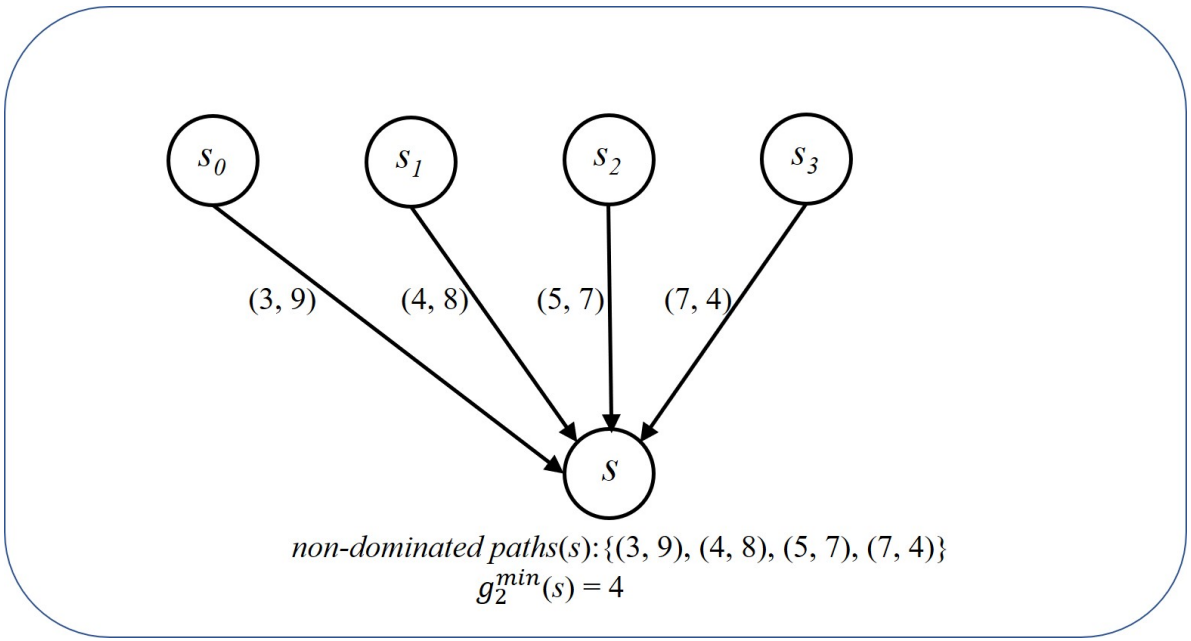
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

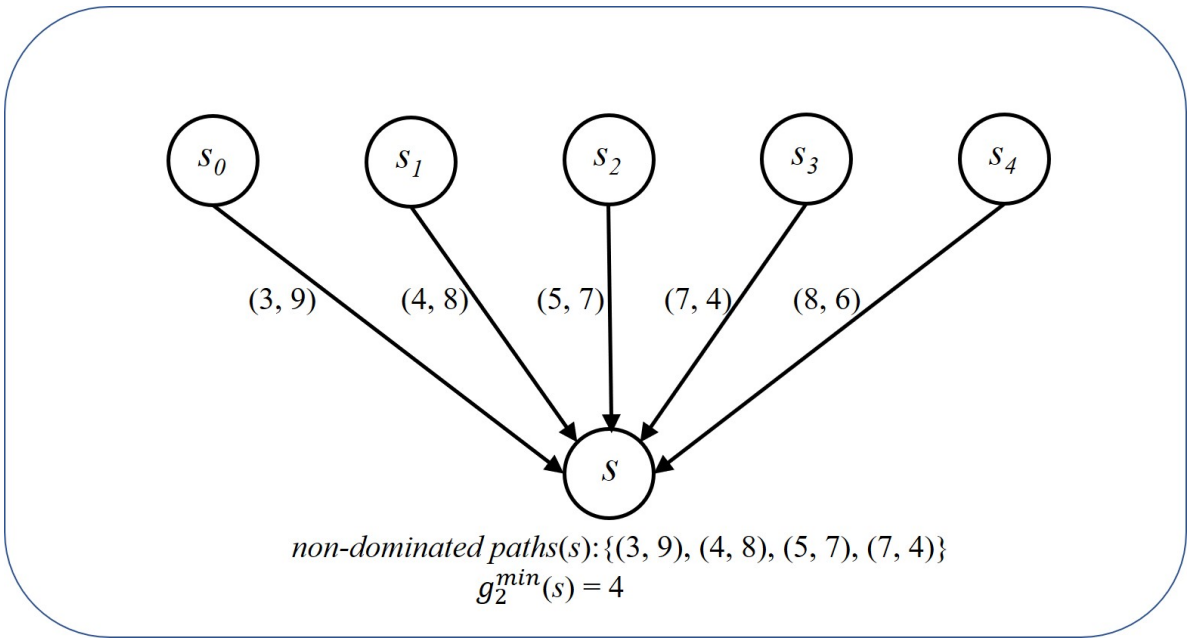
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

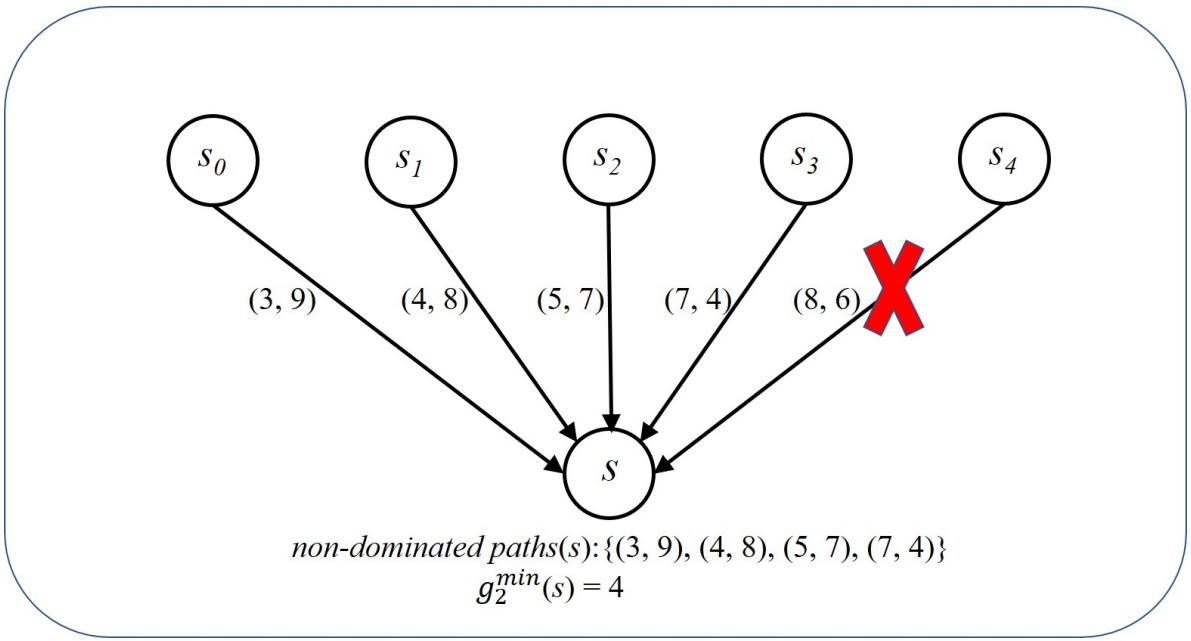
- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



Example

Bi-Objective A* (Dominance checking)

- The heuristic values h_1 and h_2 are consistent.
- The *Open* list is sorted lexicographically by (f_1, f_2).
- For each state s, BOA* maintains a $g_2^{\min}(s)$.



In BOA*, dominance check takes constant time.

Bi-Objective A*

Algorithm 2: Bi-Objective A* (BOA*)

Input : A search problem $(S, E, \mathbf{c}, s_{start}, s_{goal})$ and a consistent heuristic function $\mathbf{h}$
Output: A cost-unique Pareto-optimal solution set

1 $sols \leftarrow \emptyset$
2 **for each** $s \in S$ **do**
3 $\quad g_2^{\min}(s) \leftarrow \infty$
4 $x \leftarrow$ new node with $s(x) = s_{start}$
5 $\mathbf{g}(x) \leftarrow (0, 0)$
6 $parent(x) \leftarrow$ null
7 $\mathbf{f}(x) \leftarrow (h_1(s_{start}), h_2(s_{start}))$
8 Initialize $Open$ and add x to it
9 **while** $Open \neq \emptyset$ **do**
10 $\quad$ Remove a node x from $Open$ with the lexicographically smallest f-value of all nodes in $Open$
11 $\quad$ **if** $g_2(x) \geq g_2^{\min}(s(x)) \vee f_2(x) \geq g_2^{\min}(s_{goal})$ **then**
12 $\quad\quad$ **continue**
13 $\quad g_2^{\min}(s(x)) \leftarrow g_2(x)$
14 $\quad$ **if** $s(x) = s_{goal}$ **then**
15 $\quad\quad$ Add x to $sols$
16 $\quad\quad$ **continue**
17 $\quad$ **for each** $t \in \texttt{Succ}(s(x))$ **do**
18 $\quad\quad y \leftarrow$ new node with $s(y) = t$
19 $\quad\quad \mathbf{g}(y) \leftarrow \mathbf{g}(x) + \mathbf{c}(s(x), t)$
20 $\quad\quad parent(y) \leftarrow x$
21 $\quad\quad \mathbf{f}(y) \leftarrow \mathbf{g}(y) + \mathbf{h}(t)$
22 $\quad\quad$ **if** $g_2(y) \geq g_2^{\min}(t) \vee f_2(y) \geq g_2^{\min}(s_{goal})$ **then**
23 $\quad\quad\quad$ **continue**
24 $\quad\quad$ Add y to $Open$
25 **return** $sols$

Bi-Objective A*

Theorem

BOA computes a cost-unique Pareto-optimal solution set.*

Experimental Evaluation: Setup

- We compare to:
  - NAMOA*dr (Pulido et al., 2015)
  - BOA* with standard linear-time dominance checking (sBOA*),
  - Bi-Objective Dijkstra (BDijkstra), and Bidirectional Bi-Objective Dijkstra (BBDijkstra) (Sedeño et al., 2019).
- We use 5 road maps from the "9th DIMACS Implementation Challenge: Shortest Path".

Experimental Evaluation

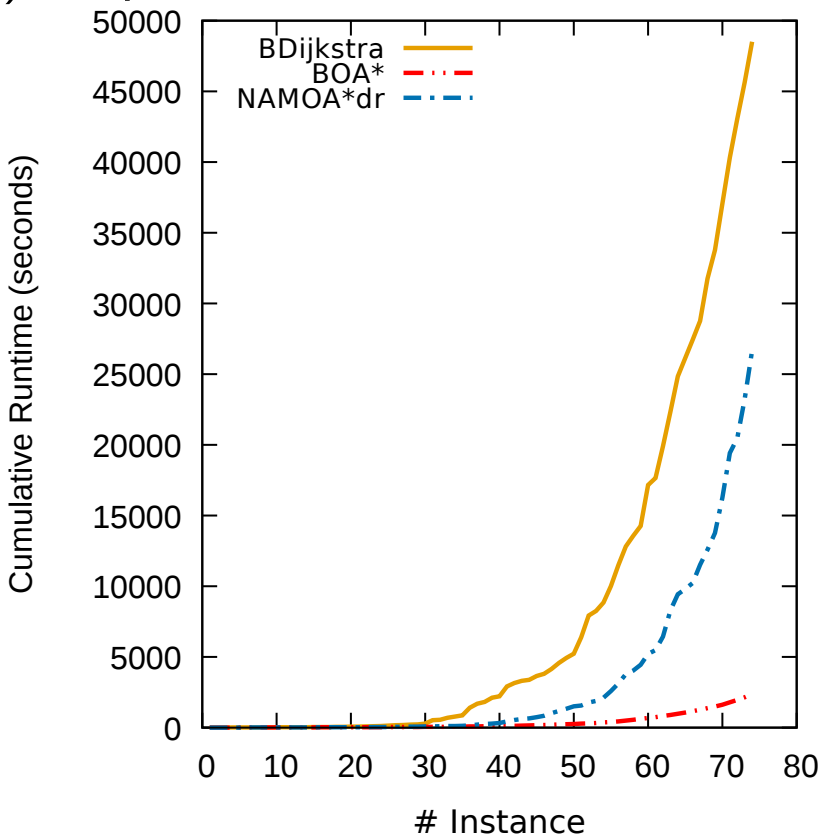
Runtime (sec) on 50 instances. After 3,600 seconds, we use 3,600 seconds in the calculation of the average.

New York City (NY)				
264,346 states, 730,100 edges, $\|sols\|$ = 199 on average				
	Solved	Average	Max	Min
NAMOA*	50/50	157.17	1,936.36	**0.02**
sBOA*	50/50	9.75	148.65	0.10
NAMOA*dr	50/50	0.65	4.99	0.11
BOA*	50/50	**0.32**	**1.95**	0.11
BBDijkstra	50/50	1.94	23.43	0.26
BDijkstra	50/50	2.55	21.16	0.17

Florida (FL)				
1,070,376 states, 2,712,798 edges, $\|sols\|$ = 739 on average				
	Solved	Average	Max	Min
NAMOA*	43/50	812.48	3,298.90	1.42
sBOA*	46/50	349.64	1,238.25	**0.43**
NAMOA*dr	50/50	19.66	329.79	**0.43**
BOA*	50/50	**4.59**	**60.54**	**0.43**
BBDijkstra	50/50	91.36	1,772.48	1.11
BDijkstra	50/50	158.33	2,722.69	0.77

Experimental Evaluation

BOA* versus the best algorithms of the state-of-the-art in the Great Lakes (LKS) map with 2,758,119 states and 6,885,658 edges.



Conclusions and Future Work

- We present BOA* a simple and fast Bi-Objective A* search algorithm.
- BOA* resembling standard A*.
- BOA* is orders of magnitude faster than state-of-the-art.
- Research directions: Bounded-optimal Bi-Objective search and Multi-Objective search.