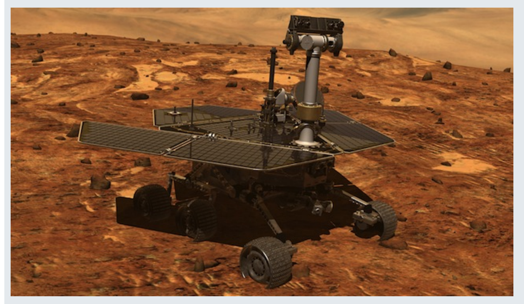
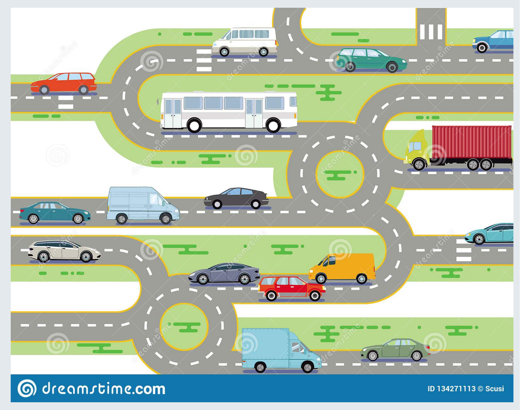
A Simple and Fast Bi-Objective Search Algorithm

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

[1]Departamento de Ciencias de la Ingeniería, Universidad Andrés Bello, Chile
[2]Department of Computer Science & Engineering, Washington University in St. Louis, USA
[3]Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile, Chile
[4]Department of Computer Science, University of Southern California, USA

Bi-objective search has many applications



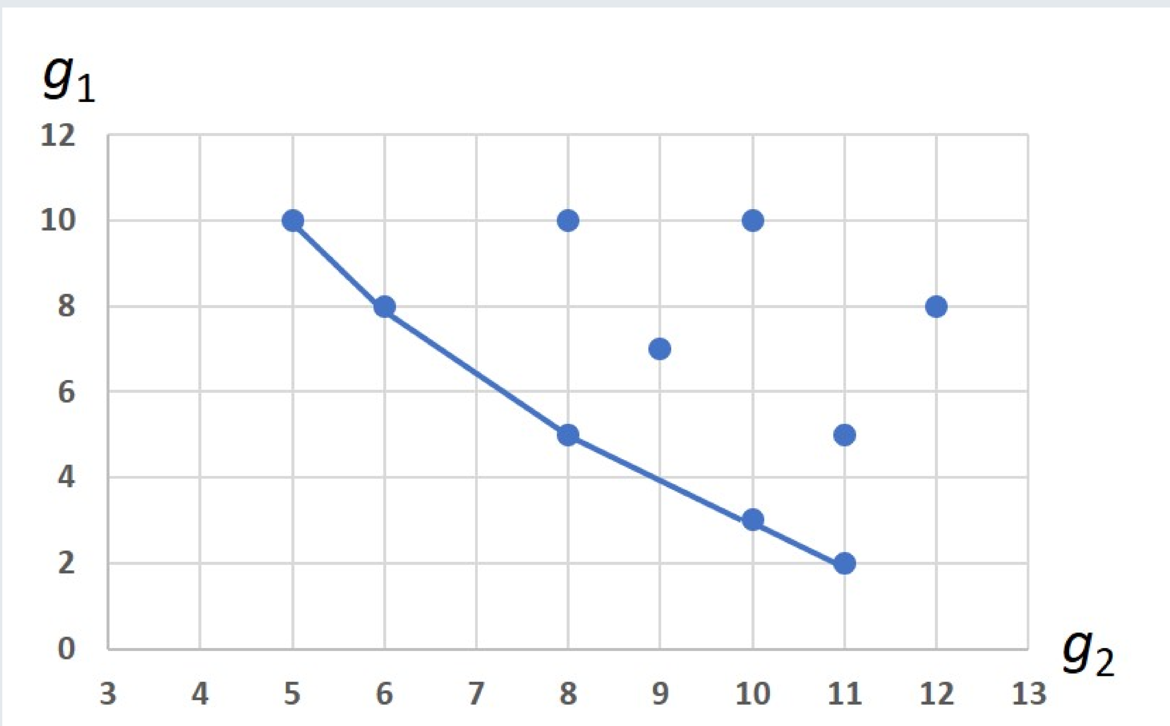


- **Path-planning in robotics**: distance and battery consumption
- **HAZMAT transport in cities**: travel time and risk of exposure for residents
- **Cycling**: distance and driver safety
- **Vehicle routing**: monetary cost and travel time

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

Bi-objective search

- **Two** objective functions C_1, C_2
- *Dominance relation*:
  $$(a,b) \prec (a',b') \text{ iff } a \le a' \text{ and } b \le b' \text{ but } (a,b) \neq (a',b')$$
- Pareto-optimal set: contains all *non-dominated* solutions

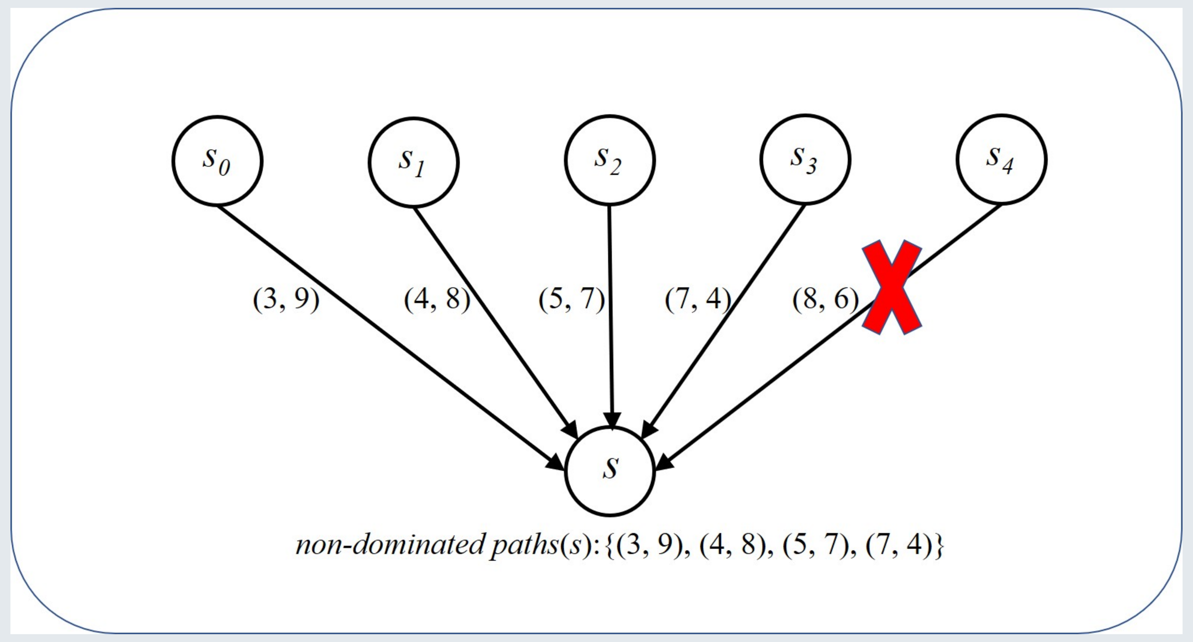


$(10,5) \prec (10,8)$, $(10,5) \not\prec (8,6)$

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.

Domination check: An example



When we find a new path to a state we need to check whether or not the cost of the newly found path is dominated/dominates previously found paths. This is a *linear-time* check. In the figure the path with cost (8,6) is dominated.

Our Contribution: Bi-Objective A*

Highlights of Bi-Objective A* (BOA*)

1. Dominance checking in constant time (instead of linear time).
2. Simple. Resembling standard A*.
3. The heuristic functions h_1 and h_2 are consistent.
4. The *Open* list is sorted lexicographically by (f_1, f_2).
5. For each state s, BOA* maintains a $g_2^{\min}(s)$.

In the example above, right before the path (8,6) is discovered, $g_2^{min}(s) = 4$ we check whether $(8,6)$ is dominated by simply evaluating $g_2^{min}(s) = 4 \le 6$

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

```
Input  : A search problem (S, E, c, s_start, s_goal) and a
         consistent heuristic function h
Output: A cost-unique Pareto-optimal solution set
sols ← ∅
for each s ∈ S do
    g2^min(s) ← ∞
x ← new node with s(x) = s_start
g(x) ← (0, 0)
parent(x) ← null
f(x) ← (h1(s_start), h2(s_start))
Initialize Open and add x to it
while Open ≠ ∅ do
    Remove a node x from Open with the
     lexicographically smallest f-value of all nodes in
     Open
    if g2(x) ≥ g2^min(s(x)) ∨ f2(x) ≥ g2^min(s_goal) then
        continue
    g2^min(s(x)) ← g2(x)
    if s(x) = s_goal then
        Add x to sols
        continue
    for each t ∈ Succ(s(x)) do
        y ← new node with s(y) = t
        g(y) ← g(x) + c(s(x), t)
        parent(y) ← x
        f(y) ← g(y) + h(t)
        if g2(y) ≥ g2^min(t) ∨ f2(y) ≥ g2^min(s_goal) then
            continue
        Add y to Open
return sols
```

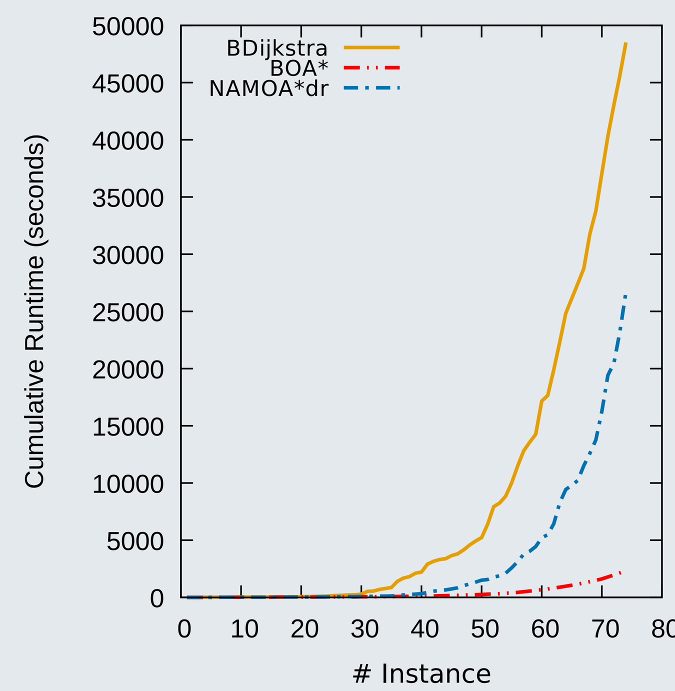
Theorem: BOA* computes a cost-unique Pareto-optimal solution set.

Experimental Evaluation

- 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".
- 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



Cumulative runtime on 74 LKS instances. The instances on the x-axis are ordered in increasing runtime of BOA*.

Conclusions and Future Work

- We present BOA* a simple and fast Bi-Objective A* search algorithm.
- BOA* resembles standard A*.
- BOA* is orders-of-magnitude faster than state-of-the-art.
- Research directions: bounded-suboptimal bi-objective search and multi-objective search.