A Simple and Fast Bi-Objective Search Algorithm

Carlos Hernández1, William Yeoh2, Jorge A. Baier3, Han Zhang4, Luis Suazo1 and Sven Koenig4

1Departamento de Ciencias de la Ingeniería, Universidad Andrés Bello, Chile
2Department of Computer Science & Engineering, Washington University in St. Louis, USA
3Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile, Chile
4Department 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: \((a, b) \prec (a', b')\) iff \( a \leq a' \) and \( b \leq b' \) but \((a, b) \neq (a', b')\)
- Pareto-optimal set: contains all non-dominated solutions

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 \( g_{min}^2(s) \).

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

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.

Experimental Evaluation

- We compare to:
  - NAMOA\textsuperscript{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)

<table>
<thead>
<tr>
<th>State</th>
<th>NAMOA\textsuperscript{dr}</th>
<th>sBOA\textsuperscript{*}</th>
<th>BOA\textsuperscript{*}</th>
<th>BBDijkstra</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLVED</td>
<td>199</td>
<td>199</td>
<td>152</td>
<td>199</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>0.02</td>
<td>0.10</td>
<td>0.11</td>
<td>0.26</td>
</tr>
<tr>
<td>MAX</td>
<td>0.23</td>
<td>0.17</td>
<td>0.10</td>
<td>0.26</td>
</tr>
<tr>
<td>MIN</td>
<td>0.23</td>
<td>0.17</td>
<td>0.10</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Florida (FL)

<table>
<thead>
<tr>
<th>State</th>
<th>NAMOA\textsuperscript{dr}</th>
<th>sBOA\textsuperscript{*}</th>
<th>BOA\textsuperscript{*}</th>
<th>BBDijkstra</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLVED</td>
<td>199</td>
<td>199</td>
<td>152</td>
<td>199</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>MAX</td>
<td>1.28</td>
<td>60.54</td>
<td>60.54</td>
<td>60.54</td>
</tr>
<tr>
<td>MIN</td>
<td>1.28</td>
<td>60.54</td>
<td>60.54</td>
<td>60.54</td>
</tr>
</tbody>
</table>

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.