Solving Domain-Independent Dynamic Programming Problems with Anytime Heuristic Search

🎉 Best Paper Award Runner-up 🎉

Ryo Kuroiwa and J. Christopher Beck
Toronto Intelligent Decision Engineering Laboratory (TIDEL)
Department of Mechanical and Industrial Engineering
University of Toronto
Recap of DIDP

Novel model-based paradigm for combinatorial optimization

Any combinatorial optimization problem

State-based DP model

DIDP solver

Current solvers are based on heuristic search

Model

Model

\[
\begin{align*}
\text{compute } V(N \setminus \{0\}, 0) \\
V(U, i) &= \min_{j \in U} c_{ij} + V(U \setminus \{i\}, j) \\
V(\emptyset, i) &= c_{i0} \\
V(U, i) &\geq 0.
\end{align*}
\]
Recap of DIDP

compute $V(N \setminus \{0\}, 0, 0)$

$$V(U, i, t) = \begin{cases} 
\min_{j \in U: t + c_{ij} \leq b_j} c_{ij} + V(U \setminus \{i\}, j, \max\{t + c_{ij}, a_j\}) & \text{if } U \neq \emptyset \\
0 & \text{else if } i \neq 0 \\
c_{ij} + V(\emptyset, 0, t + c_{i0}) & \text{otherwise}
\end{cases}$$

$V(U, i, t) \leq V(U, i, t')$ if $t \leq t'$

$V(U, i, t) \geq 0.$
Prototype Solver: CAASDy

- Solve DP as a shortest path in the state space using A*
- Heuristic: dual bound defined in a DP model

Implemented in [https://github.com/domain-independent-dp/didp-rs](https://github.com/domain-independent-dp/didp-rs)
Prototype Solver: CAASDy

- Solve **DP as a shortest path** in the state space using A*
- **Heuristic:** dual bound defined in a DP model

No solution until optimality solved

Implemented in [https://github.com/domain-independent-dp/didp-rs](https://github.com/domain-independent-dp/didp-rs)
Anytime Solvers

- Quickly find a solution and continuously improve it
- Standard in OR (e.g., MIP and CP)

Can we develop anytime solvers for DIDP?
## Anytime Heuristic Search Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth First Branch-and-Bound (DFBnB)</td>
<td>DFS</td>
<td></td>
</tr>
<tr>
<td>Cyclic Best-First Search (CBFS)</td>
<td>Hybrid of DFS and best-first search</td>
<td>Kao et al. 2009</td>
</tr>
<tr>
<td>Anytime Column Progressive Search (ACPS)</td>
<td>Hybrid of DFS and beam search</td>
<td>Vadlamudi et al. 2012</td>
</tr>
<tr>
<td>Discrepancy-Bounded DFS (DBDFS)</td>
<td>Discrepancy-based</td>
<td>Beck and Perron 2000</td>
</tr>
<tr>
<td>Complete Anytime Beam Search (CABS)</td>
<td>Iterative beam search</td>
<td>Zhang 1998</td>
</tr>
</tbody>
</table>

Implemented in [https://github.com/domain-independent-dp/didp-rs](https://github.com/domain-independent-dp/didp-rs)
# Anytime Heuristic Search Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth First Branch-and-Bound (DFBnB)</td>
<td>DFS</td>
<td></td>
</tr>
<tr>
<td>Cyclic Best-First Search (CBFS)</td>
<td>Hybrid of DFS and best-first search</td>
<td>Kao et al. 2009</td>
</tr>
<tr>
<td>Anytime Column Progressive Search (ACPS)</td>
<td>Hybrid of DFS and beam search</td>
<td>Vadlamudi et al. 2012</td>
</tr>
<tr>
<td>Discrepancy-Bounded DFS (DBDFS)</td>
<td>Discrepancy-based</td>
<td>Beck and Perron 2000</td>
</tr>
<tr>
<td>Complete Anytime Beam Search (CABS)</td>
<td>Iterative beam search</td>
<td>Zhang 1998</td>
</tr>
</tbody>
</table>

Implemented in [https://github.com/domain-independent-dp/didp-rs](https://github.com/domain-independent-dp/didp-rs)
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

\[
\{1, 2, 3\}, 0, 0
\]

$f: 0$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

$\{1, 2, 3\}, 0, 0$

$f: 0$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

{2, 3}, 1, 5

{1, 2, 3}, 0, 0

{1, 3}, 2, 4

{1, 2}, 3, 8

f: 3

f: 4

f: 5
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- $\{2, 3\}, 1, 5$
  - $f: 3$

- $\{1, 2, 3\}, 0, 0$
  - $f: 4$

- $\{1, 3\}, 2, 4$
  - $f: 4$

- $\{1, 2\}, 3, 8$
  - $f: 5$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- $\{2, 3\}, 1, 5$
  - $f: 8$
- $\{3\}, 2, 10$
  - $f: 8$
- $\{2\}, 3, 9$
  - $f: 7$
- $\{1, 3\}, 2, 4$
  - $f: 9$
- $\{3\}, 1, 9$
  - $f: 9$
- $\{1\}, 3, 8$
  - $f: 7$
- $\{1, 2\}, 3, 8$
  - $f: 7$

$\{1, 2, 3\}, 0, 0$
- $f: 7$

$\{1\}, 3, 8$
- $f: 7$

$\{1\}, 3, 8$
- $f: 7$

$\{1\}, 3, 8$
- $f: 7$

$\{1\}, 3, 8$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- $\{2, 3\}, 1, 5$
- $\{3\}, 2, 10$
- $\{2\}, 3, 9$
- $\{3\}, 1, 9$
- $\{3\}, 1, 9$
- $\{1\}, 3, 8$
- $\emptyset, 1, 12$

No transitions

$f: 11$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

No transitions
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- {2, 3}, 1, 5
  - {3}, 2, 10
  - {2}, 3, 9

- {1, 3}, 2, 4
  - {3}, 1, 9
  - {1}, 3, 8
    - $\emptyset$, 1, 12
      - $\emptyset$, 0, 15

- {1, 2, 3}, 0, 0
  - 4
  - 5

- {1, 2}, 3, 8
  - 3
  - 4

$f$: 14
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- $\{2, 3\}, 1, 5$
- $\{3\}, 2, 10$
- $\{2\}, 3, 9$
- $\{3\}, 1, 9$
- $\{1, 3\}, 2, 4$
- $\{3\}, 1, 9$
- $\{1\}, 3, 8$
- $\emptyset, 1, 12$
- $\emptyset, 1, 12$
- $\emptyset, 0, 15$

$f: 14$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- $\{2, 3\}, 1, 5$
- $\{3\}, 2, 10$
- $\{2\}, 3, 9$
- $\{3\}, 1, 9$
- $\{1, 3\}, 2, 4$
- $\{3\}, 1, 9$
- $\{1\}, 3, 8$
- $\emptyset, 1, 12$
- $\emptyset, 0, 15$
- $\{1\}, 3, 8$
- $\{1, 2\}, 3, 8$
- $\{1, 2, 3\}, 0, 0$

$f: 14$
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

- Finishing time (makespan)
- Travel time (w/o waiting)
Beam Search

- Keep $k$ best states according to the $f$-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

Beam search can reduce the memory usage by keeping only states in the current layer

- Finishing time (makespan)
- Travel time (w/o waiting)
**Complete Anytime Beam Search (CABS)**

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state $s$ if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

$\{1, 2, 3\}, 0, 0$

$f$: 0

Zhang 1998
Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state $s$ if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

\[
\begin{align*}
\{2, 3\}, & \quad 1, 5 \quad f: 3 \\
\{1, 2, 3\}, & \quad 0, 0 \quad f: 4 \\
\{1, 2\}, & \quad 3, 8 \quad f: 5
\end{align*}
\]
Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state $s$ if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

Zhang 1998
Complete Anytime Beam Search (CABS)

- Beam search with \( k = 1, 2, 4, 8, 16, \ldots \) until states are exhausted
- Prune a state \( s \) if \( f(s) \geq \) the incumbent solution cost

\[ \begin{array}{ccc}
\{1\} & 3 & 8 \\
\{2\} & 3 & 9 \\
\{3\} & 1 & 9 \\
\emptyset & 1 & 12 \\
\{1, 2\}, 3 & 8 \\
\{2, 3\}, 1 & 5 \\
\{3\} & 2 & 10 \\
\emptyset, 3 & 13 \\
\end{array} \]

\[ \begin{array}{ccc}
\{1, 2, 3\} & 0 & 0 \\
\{1, 3\} & 2 & 4 \\
\emptyset & 1 & 12 \\
\{1\} & 3 & 8 \\
\{2\} & 3 & 9 \\
\{3\} & 1 & 9 \\
\emptyset, 3 & 13 \\
\end{array} \]

\( k = 8, \text{ incumbent: 14} \)

\( f: 11 \)
Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state $s$ if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

Zhang 1998
Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state $s$ if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

Zhang 1998
Experimental Evaluation
Primal gap: $\frac{\text{solution cost} - \text{best known cost}}{\text{solution cost}}$ (1 if no solution found)

Primal Integral

Primal gap: $\frac{\text{solution cost} - \text{best known cost}}{\text{solution cost}}$ (1 if no solution found)
The other anytime methods are between CABS and MIP/CP.
Heuristic search algorithms except for CABS reach 8GB memory limit.
Coverage and Gap (TSPTW)

- Ratio of instances
- Time to solve optimally
- Primal gap at limit
- MIP
- CAASDy
- CABS
- CP (coverage < 0.4)
Coverage and Gap \((1\|\sum w_i T_i)\)

- Coverage and Gap
- Ratio of instances
- Time to solve optimally
- Primal gap at limit
- CP
- CABS
- CAASDy
- MIP (coverage < 0.4)
## Coverage in Each Problem

<table>
<thead>
<tr>
<th>Description</th>
<th>MIP</th>
<th>CP</th>
<th>CAASDy</th>
<th>CABS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSPTW (340)</td>
<td>227</td>
<td>47</td>
<td>257</td>
<td>259</td>
</tr>
<tr>
<td>TSP with time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVRP (207)</td>
<td>26</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>vehicle routing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALBP-1 (2100)</td>
<td>1357</td>
<td>1584</td>
<td>1653</td>
<td>1801</td>
</tr>
<tr>
<td>assembly line</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bin Packing (1615)</td>
<td>1157</td>
<td>1234</td>
<td>922</td>
<td>1163</td>
</tr>
<tr>
<td>bin packing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOSP (570)</td>
<td>225</td>
<td>437</td>
<td>483</td>
<td>527</td>
</tr>
<tr>
<td>manufacturing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph-Clear (135)</td>
<td>24</td>
<td>4</td>
<td>76</td>
<td>103</td>
</tr>
<tr>
<td>building security</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Talent Scheduling (1000)</td>
<td>6</td>
<td>7</td>
<td>224</td>
<td>253</td>
</tr>
<tr>
<td>scheduling actors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m-PDTSP (1117)</td>
<td>945</td>
<td>1049</td>
<td>947</td>
<td>1035</td>
</tr>
<tr>
<td>pick up &amp; delivery</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>∑wiTi (375)</td>
<td>109</td>
<td>150</td>
</tr>
<tr>
<td>job scheduling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# of optimality solved instances with 8GB and 30-min
# Mean Primal Gap and Primal Integral

<table>
<thead>
<tr>
<th>Description</th>
<th>MIP</th>
<th>CP</th>
<th>CABS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSPTW (340)</td>
<td>0.227/484.05</td>
<td>0.026/48.97</td>
<td>0.003/8.97</td>
</tr>
<tr>
<td>CVRP (207)</td>
<td>0.585/1157.43</td>
<td>0.317/601.15</td>
<td>0.185/351.21</td>
</tr>
<tr>
<td>SALBP-1 (2100)</td>
<td>0.345/634.64</td>
<td>0.005/28.48</td>
<td>0.000/1.92</td>
</tr>
<tr>
<td>Bin Packing (1615)</td>
<td>0.039/86.19</td>
<td>0.002/8.04</td>
<td>0.002/5.26</td>
</tr>
<tr>
<td>MOSP (570)</td>
<td>0.039/100.41</td>
<td>0.004/13.01</td>
<td>0.000/0.36</td>
</tr>
<tr>
<td>Graph-Clear (135)</td>
<td>0.110/311.83</td>
<td>0.015/44.27</td>
<td>0.000/0.49</td>
</tr>
<tr>
<td>Talent Scheduling (1000)</td>
<td>0.051/142.69</td>
<td>0.002/18.14</td>
<td>0.011/26.36</td>
</tr>
<tr>
<td>m-PDTSP (1178)</td>
<td>0.078/180.00</td>
<td>0.013/26.04</td>
<td>0.002/5.33</td>
</tr>
<tr>
<td>(1|\sum w_i T_i) (375)</td>
<td>0.018/74.56</td>
<td>0.000/2.26</td>
<td>0.034/73.60</td>
</tr>
</tbody>
</table>

Mean primal gap at limit / primal integral
Conclusion

- Anytime DIDP solvers are promising!
- Trade-off between time and memory
- Future work: parallelization?

![Diagram showing solvers by anytime and multi-threaded vs. single-threaded](image-url)
Please Use DIDP!

We need your ideas to advance DIDP!

● Visit our website: https://didp.ai

● Start DIDP with Python: pip install didppy
   Tutorials and API Reference: https://didppy.rtfd.io

● Start DIDP with YAML: cargo install didp-yaml

● Clone the repository:
  git clone https://github.com/domain-independent-dp/didp-rs

Everything in Rust 🦀
Why Not Anytime Weighted A*?

- A user may provide 0 dual bound (heuristic)
- Finding a satisficing solution is usually much easier in combinatorial optimization than in AI planning