Domain-Independent Dynamic Programming

Ryo Kuroiwa and J. Christopher Beck
Department of Mechanical and Industrial Engineering, University of Toronto

TIDEL Toronto Intelligent Decision Engineering Laboratory

UNIVERSITY OF TORONTO

Research Question

Can we use **dynamic programming as a model-based paradigm** for combinatorial optimization?

Developed Software

Use our software to **solve your problem by just defining a DP model!**

We developed general-purpose **heuristic search solvers!**

Install the Python interface: `pip install didppy`

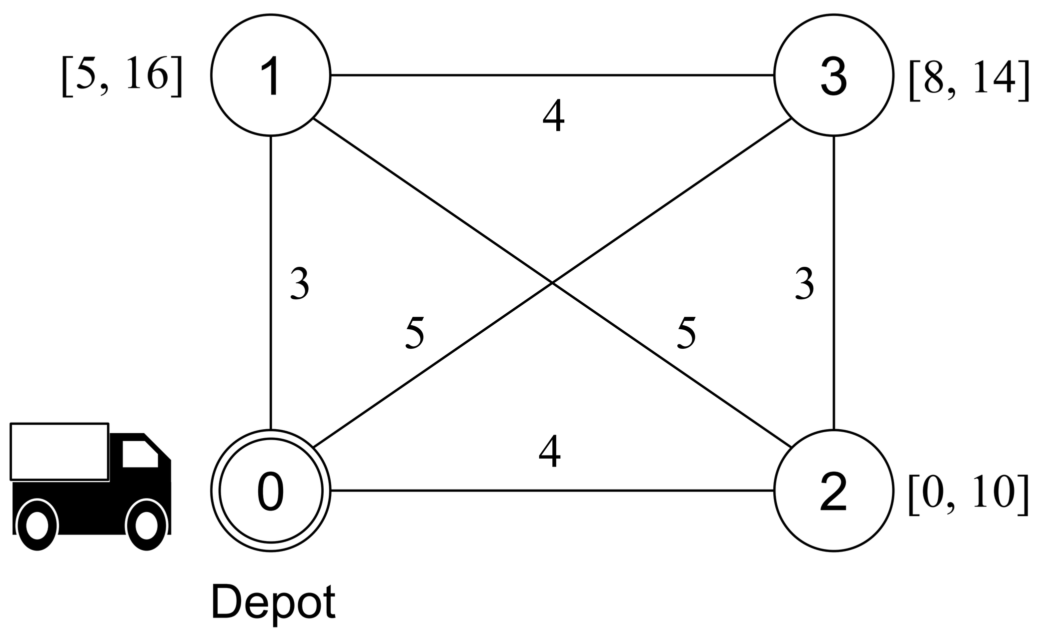
Open-source and free for commercial use (MIT/Apache-2.0).

Project page

Tutorials, examples, and API reference

TSPTW Example

Minimize the travel time of a tour to visit all customers within the time windows.



DP Model for TSPTW

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

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

State variables:
- U: unvisited customers
- i : current customer
- t : current time

Constants
- N: all customers (0: depot)
- $[a_i, b_i]$: time window for customer i
- c_{ij}: travel time from customer i to j

Modeling and Solving in DIDP

```python
import didppy as dp

model = dp.Model(maximize=False)

customer = model.add_object_type(number=4)
a = [0, 5, 0, 8]
b = [100, 16, 10, 14]
c = model.add_int_table([[0, 3, 4, 5], [3, 0, 5, 4], [4, 5, 0, 3], [5, 4, 3, 0]])

u = model.add_set_var(object_type=customer, target=[1, 2, 3])
i = model.add_element_var(object_type=customer, target=0)
t = model.add_int_resource_var(target=0, less_is_better=True)

for j in range(1, 4):
    visit = dp.Transition(
        name="visit {}".format(j),
        cost=c[i, j] + dp.IntExpr.state_cost(),
        effects=[(u, u.remove(j)), (i, j), (t, dp.max(t + c[i, j], a[j]))],
        preconditions=[u.contains(j), t + c[i, j] <= b[j]],
    )
    model.add_transition(visit)

return_to_depot = dp.Transition(
    name="return",
    cost=c[i, 0] + dp.IntExpr.state_cost(),
    effects=[(i, 0), (t, t + c[i, 0])],
    preconditions=[u.is_empty(), i != 0],
)
model.add_transition(return_to_depot)

model.add_base_case([u.is_empty(), i == 0], cost=0)

for j in range(1, 4):
    model.add_state_constr(~u.contains(j) | (t + c[i, j] <= b[j]))

model.add_dual_bound(0)

solver = dp.CABS(model)
solution = solver.search()
```

What DIDP Can Do but PDDL Cannot

Explicitly modeling implications of the problem definition that can be useful solvers (common in OR!).

State constraints (can be used for pruning)

$V(U,i,t) = \infty$ if $\exists j \in U, t + c_{ij} > b_j$

```python
for j in range(1, 4):
    model.add_state_constr(~u.contains(j) | (t + c[i, j] <= b[j]))
```

Dominance with **resource variables** (can be used for pruning)

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

```python
t = model.add_int_resource_var(target=0, less_is_better=True)
```

Dual bound (can be used as a heuristic)

$V(U,i,t) \geq 0$

```python
model.add_dual_bound(0)
```

Heuristic Search Solvers

Heuristic search solves a DP model as a shortest path problem in a **state space** using the dual bound as a heuristic.

We developed the following solvers:

- CAASDy: A*.
- CABS: performs beam search with exponentially increasing beam width (anytime and complete).
- 5 other anytime heuristic search solvers.

Promising performance compared to MIP and CP.

Future work: parallelization, domain-independent dual bound

	Description	MIP	CP	CAASDy	CABS
TSPTW (340)	TSP with time	227/0.227	47/0.026	257/0.244	**259/0.003**
CVRP (207)	vehicle routing	**26**/0.585	0/0.317	5/0.976	6/**0.185**
SALBP-1 (2100)	assembly line	1357/0.345	1584/0.005	1653/0.213	**1801/0.000**
Bin Packing (1615)	bin packing	1157/0.039	**1234/0.002**	922/0.429	1163/**0.002**
MOSP (570)	manufacturing	225/0.039	437/0.004	483/0.153	**527/0.000**
Graph-Clear (135)	building security	24/0.110	4/0.015	76/0.437	**103/0.000**
Talent Scheduling (1000)	scheduling actors	6/0.051	7/**0.002**	224/0.793	**253**/0.011
m-PDTSP (1117)	pick up & delivery	945/0.078	**1049**/0.013	947/0.196	1035/**0.002**
$1\|\|\sum w_i T_i$ (375)	job scheduling	109/0.018	150/**0.000**	270/0.280	**285**/0.034

Coverage / primal gap (gap to the best known cost) achieved within 8GB and 30-min.