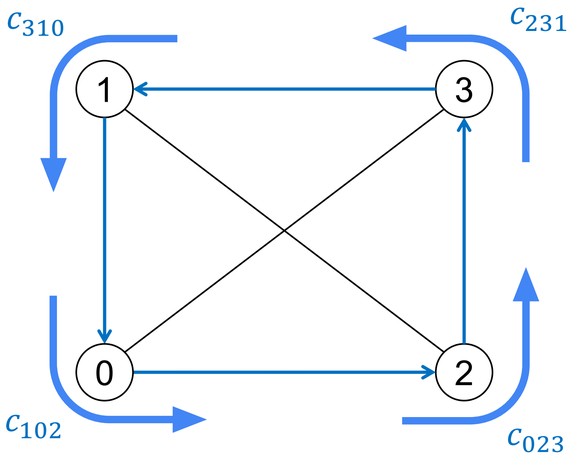
New Exact Methods for Solving Quadratic Traveling Salesman Problem

Yuxiao (Jasper) Chen[1], Anubhav Singh[1], Ryo Kuroiwa[2], and J. Christopher Beck[1]
[1]University of Toronto, Canada
[2]National Institute of Informatics / SOKENDAI, Japan

Quadratic Traveling Salesman Problem

- Minimize the cost of a tour visiting all locations
- A cost is assigned to each triplet of consecutive visits

Cost : $c_{023} + c_{231} + c_{310} + c_{102}$



Motivations:
- Robotics: a smooth trajectory with small turning angles
  [Aggrawal et al. 2000; Salva et al. 2008]
- Bioinformatics: a permutation maximizing the log likelihood of a statistical model related to DNA
  [Jäger and Molitor 2008; Fischer et al. 2014]

Constraint Programming (CP)

CP-AD: based on all-different

$$\min\ c_{x_{n-1}x_0x_1} + \sum_{i=0}^{n-3} c_{x_ix_{i+1}x_{i+2}} + c_{x_{n-2}x_{n-1}x_0}$$
$\text{all\_different}(x_0, \dots, x_{n-1})$ Each location is visited once
$x_0 = 0$ 0 is in the 0-th position wlog
$x_i \in N \quad i = 0, \dots, n-1$ Location x_i is in the i-th position

CP-CIR: based on circuit

$$\min\ \sum_{i \in N} \sum_{j \in N \setminus \{i\}} \sum_{k \in N \setminus \{i,j\}} c_{ijk}\, x_{ij} x_{jk}$$
$\text{circuit}(z_0, \dots, z_{n-1})$ z forms a circuit
$x_{ij} \leftrightarrow (z_i = j)\ \forall i \in N, j \in N \setminus \{i\}$ Visit j from i
$z_i \in N \setminus \{i\} \quad \forall i \in N$ Location z_i is visited after location i

Mixed-Integer Linear Programming (MILP)

$$\min\ \sum_{i \in N} \sum_{j \in N \setminus \{i\}} \sum_{k \in N \setminus \{i,j\}} c_{ijk} y_{ijk}$$

$\sum_{j \in N \setminus \{i\}} x_{ij} = \sum_{j \in N \setminus \{i\}} x_{ji} = 1$	$\forall i \in N$	Each location is visited once
$x_{ij} = \sum_{k \in N \setminus \{i,j\}} y_{ijk} = \sum_{k \in N \setminus \{i,j\}} y_{kij}$	$\forall i \in N, j \in N \setminus \{i\}$	Link x and y
$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2$	$\forall i \in N \setminus \{0\}, j \in N \setminus \{i, 0\}$	$u_j \geq u_i + 1$ if $x_{ij} = 1$
$1 \leq u_i \leq n-1$	$\forall i \in N \setminus \{0\}$	Position of location i in the tour
$x_{ij} \in \{0,1\}$	$\forall i \in N, j \in N \setminus \{i\}$	Visit j from i
$y_{ijk} \in \{0,1\}$	$\forall i \in N, j \in N \setminus \{i\}, k \in N \setminus \{i,j\}$	Visit i, j, and k consecutively

Variations:
- Mixed-Integer Quadratic Programming (**MIQP**): replace y_{ijk} with $x_{ij}x_{jk}$ in the objective
- Branch-and-Cut (**B&C**): remove u_i and lazily generate subtour elimination constraints
  [Fischer et al. (2014), Oswin et al. (2017)]

Domain-Independent Dynamic Programming (DIDP)

Project page: https://didp.ai

- State-based model & solve framework **inspired by PDDL planning**
- A dynamic programming model is solved by a **general-purpose heuristic search solver**

State (U, i, t, f):
- U: the set of unvisited locations
- i: the previous location
- j: the current location
- f: the first location visited after 0

$$\text{compute } V(N \setminus \{0\}, 0, 0, 0)$$ Initial state (no location is visited except for 0)

$$V(U,i,j,f) = \begin{cases} \min_{k \in U} V(U \setminus \{k\}, 0, k, k) & \text{if } j = 0 \\ \min_{k \in U} c_{ijk} + V(U \setminus \{k\}, j, k, f) & \text{if } j \neq 0 \wedge U \neq \emptyset \\ c_{j0f} + c_{ij0} & \text{if } U = \emptyset \end{cases}$$

Action (visit the first location k) / Action (visit a location k) / Goal (all locations are visited)

In DIDP, a lower bound on the solution cost can be **explicitly modeled** and used as an **admissible heuristic**

DIDP-1: $O(n)$, two similar bounds with c_{lkm} and c_{klm} also used

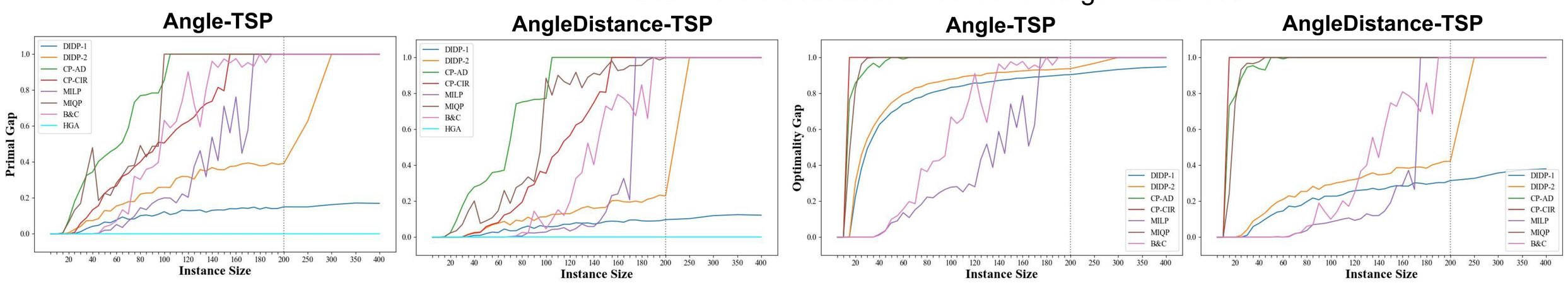
$$V(U,i,j,f) \geq \sum_{k \in U \cup \{f,0\}} \min_{l \in N \setminus \{k\}, m \in N \setminus \{k,l\}} c_{lmk}$$

DIDP-2: $O(n^2)$, two similar bounds with c_{lkm} and c_{klm} also used

$$V(U,i,j,f) \geq \sum_{k \in U \cup \{f,0\}} \max_{e \in N \setminus (U \cup \{0,i,j\})} \min_{l \in N \setminus \{k,e\}, m \in N \setminus \{k,l,e\}} c_{lmk}$$

Experimental Evaluation

- B&C and MILP are competitive on small instances
- DIDP is the best exact method for larger instances



- **Angle-TSP** [Aggrawal et al. 2000]: minimize the sum of turning angles, **AngleDistance-TSP** [Salva et al. 2008]: the weighted sum of turning angles and distance
- **Primal gap**: normalized solution cost (1: no solution), **Optimality gap**: relative difference between lower and upper bounds (0: optimally solved, 1: no solution)
- MILP, MIQP, and B&C: Gurobi 11.0.2, CP: CP Optimizer 22.1.0.0, DIDP: CABS in didp-rs 0.8.0, 1 thread, 30 min, 8GB RAM for ~200 locations, 16 GB for more