Additive bounding for the Double TSP with Multiple Stacks

Luca Diedolo, ACT-OR
luca.diedolo@ahead-research.com

Giovanni Righini, University of Milan
giovanni.righini@unimi.it

Optimization and Decision Science Conference 2020

November 19^{th}, 2020

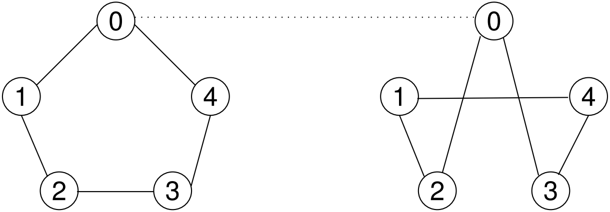
The DTSPMS

The Double Traveling Salesman Problem with Multiple Stacks is an NP-*hard* combinatorial optimization problem.

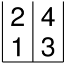
- Data:
  - two weighted graphs (*pickup* and *delivery*) with a depot each
  - a vehicle with a given number of stacks (LIFO policy)
- solution: a Hamiltonian cycle for each graph, based at the depot
- objective: minimize the total cost of the two cycles
- constraints: the two cycles must be compatible with the LIFO policy for each stack: picked up items are put on top of a stack and only items on top of a stack can be delivered.

A sample instance

$n = 4$



pickup delivery



$l = 2$

$s = 2$

Figure: A sample instance with $n = 4$, $s = 2$, $l = 2$

State of the art

Only very small instances have been solved to optimality.

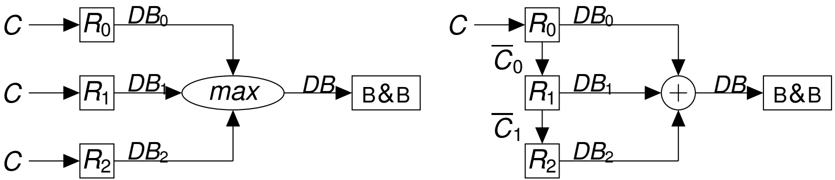
- Petersen, Archetti, Speranza (2010): branch-and-cut on a two-index vehicle flow formulation with additional infeasible path constraints;
- Carrabs, Cerulli, Speranza (2013): branch-and-bound for the DTSPMS with 2 stacks;
- Alba Martínez, Cordeau, Dell'Amico, Iori (2013): branch-and-cut: up to $n = 28$ nodes in one hour.

Our goal

- testing the tightness of additive bounds (Fischetti, Toth, 1989);
- using them in a branch-and-bound algorithm.

Additive Bounds

Multiple bounds **Additive bounds**

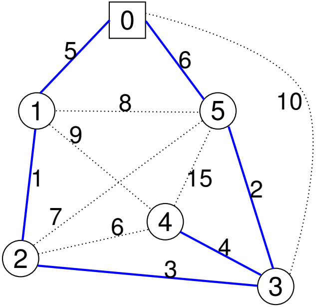


- $C \rightarrow$ original costs
- $\overline{C} \rightarrow$ reduced costs
- $R \rightarrow$ relaxations
- $DB \rightarrow$ dual bounds

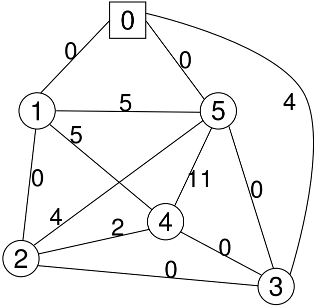
Routing step

- *Held and Karp* method to compute TSP lower bound
  - *primal* step – compute 1-*tree*
  - *dual* step – modify edge costs using *subgradient optimization*

1-*tree* computed by HK



the corresponding *residual instance*



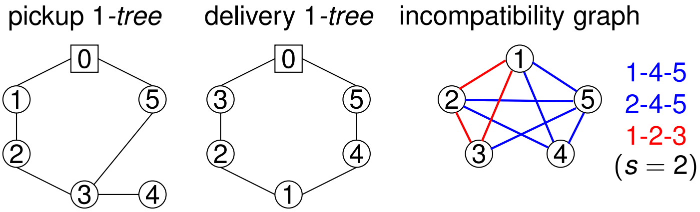
TSP optimal cost: 24

1-*tree* cost: 21 (HKLB)

Listing step

Incompatibility graph, from an arbitrary orientation of the two cycles:

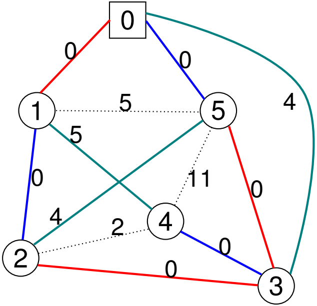
- STRAIGHT edge: $i \prec j$ in both pickup and delivery cycles
- REVERSE edge: $i \prec j$ in pickup cycle and $j \prec i$ in delivery cycle



Find all cliques of cardinality $s + 1$, where s is the number of stacks.

Repair step

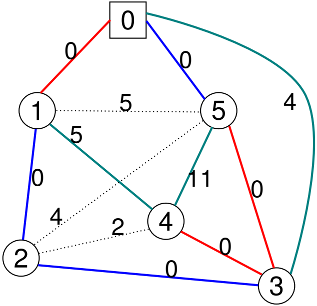
Destructive



• + • before repair

- minimum cost *subtour* of clique **1-4-5** of cost **13** (0-3-4-1-2-5-0)
- similarly for **1-2-3**, cost **9**

Non-destructive

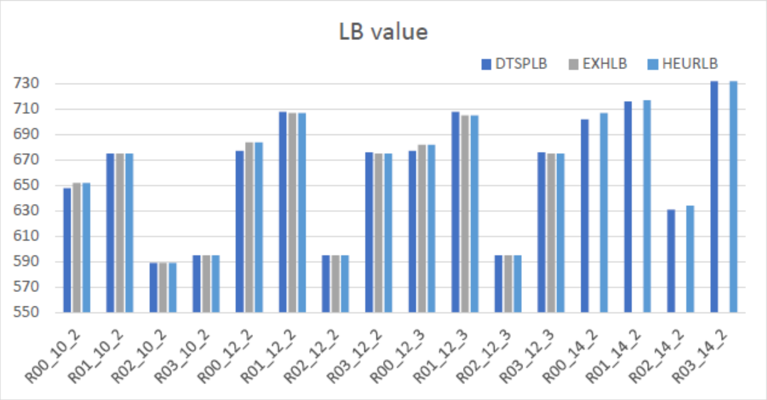


• + • after repair

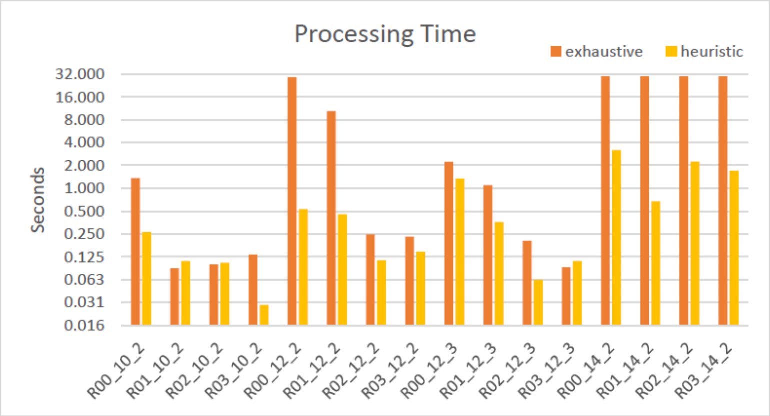
- minimum cost *subtour* of pair (**1-4-5**, **1-2-3**) of cost **20** (0-3-2-1-4-5-0)

Non-destructive repair cost: exact or heuristic.

Computational results: lower bounds

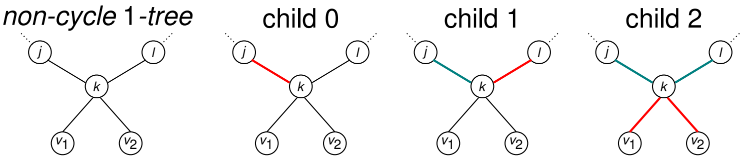


Computational results: computing time



Branching

- 1-*tree* branching ($DEG(k) > 2$)
  - *forbidden* and *fixed* edges

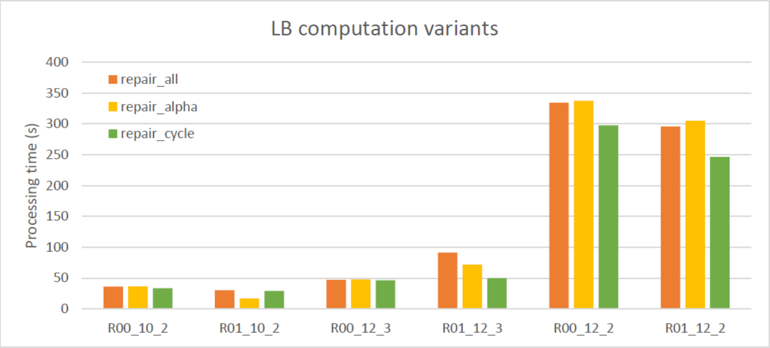


- *repairing subtour* branching
  - similarly, only with *positive* reduced cost edges

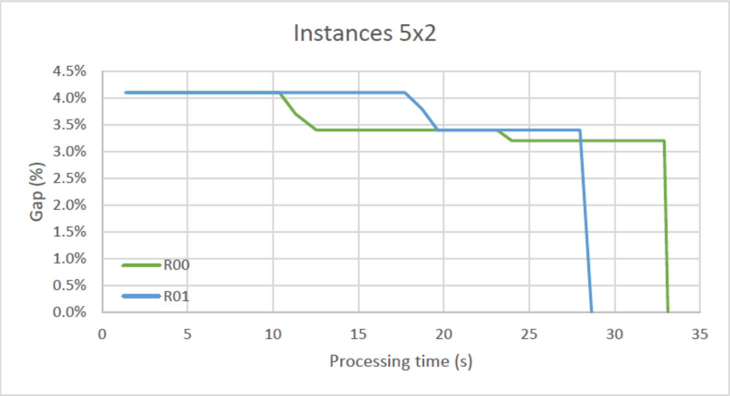
Bounding

Three variants tested for computing the additive LB

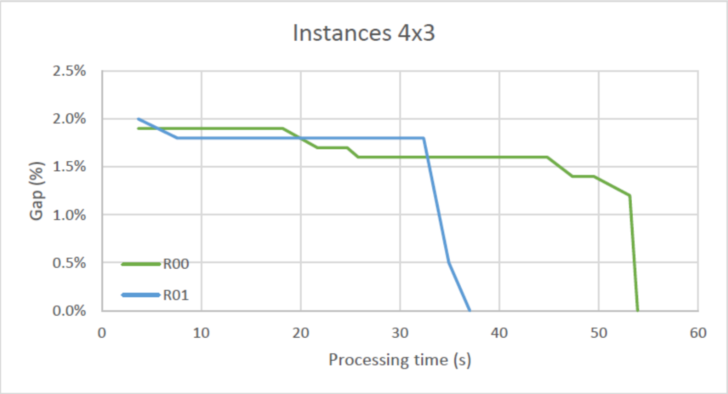
- `repair_all` – repair is always performed
- `repair_alpha` – repair is performed only when $\alpha = \frac{UB-LB}{UB}$ is small enough
- `repair_cycle` – repair is performed only when 1-*tree*s are cycles



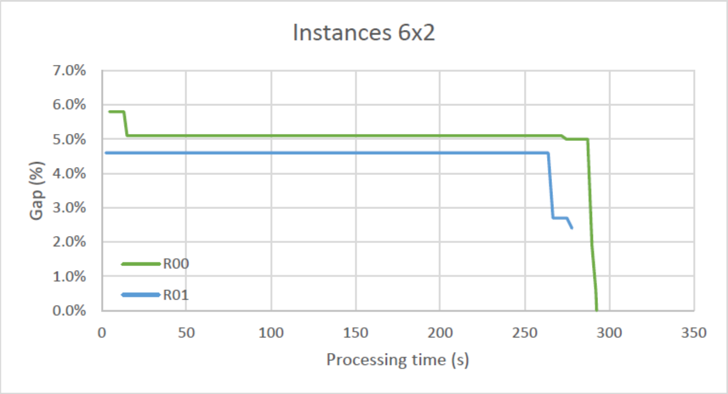
DTSPMS B&B algorithm – Experiments (5 × 2)



DTSPMS B&B algorithm – Experiments (4×3)



DTSPMS B&B algorithm – Experiments (6×2)



Conclusions

Conclusions

- additive lower bounds exceed the double TSP bound in about 75% of instances;
- *heuristic non-destructive* cost computation is very effective;
- `repair_cycle` saves processing time;
- results are still far from state-of-the-art.

Future developments

- Improve the computation of the Held-Karp lower bound;
- combinatorial explosion due to combinatorial number of checks $\rightarrow$ develop a **heuristic** also for *destructive* repair costs.