A comparative study of branch-and-price algorithms for vehicle routing with time windows and waiting time costs

Stefano Michelini[1], Yasemin Arda[1], Hande Küçükaydın[2]

[1]HEC Liège, Management School of the University of Liège

[2]Department of Industrial Engineering, MEF University, Istanbul

July 10, 2018

Overview

- Problems under consideration:
  - The VRP with Time Windows (VRPTW)
  - The VRPTW with Waiting Time Costs (VRPTWWTC)
- Branch-and-Price (BP): an effective exact solution method for the VRPTW
- Solving the NP-hard pricing problem efficiently is crucial for BP effectiveness
- Labeling algorithms are the leading methodology for solving the pricing problem

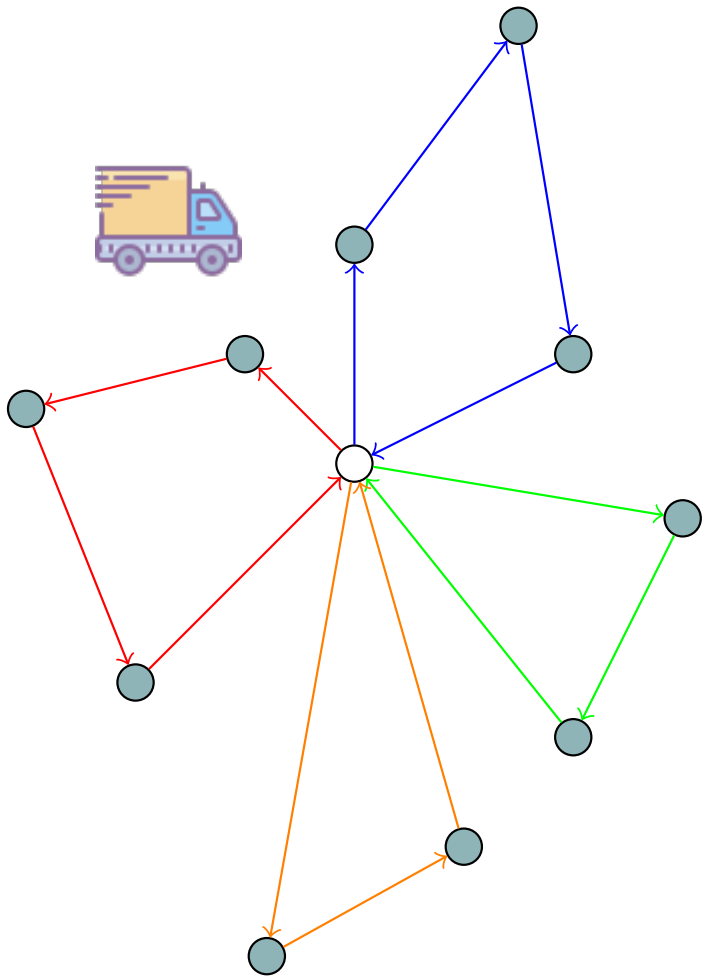
Objective

Rigorously compare the performance of several labeling algorithms in a BP framework for the VRPTW and the VRPTWWTC.

Table of Contents

Table of Contents

VRPTW: Problem definition



- Goal: find a least-cost set of routes servicing all customers
- Defined on a directed graph $G = (V, A)$
- $V := \{0, 1, \ldots, n, n+1\} = N \cup \{0, n+1\}$
- For each customer i:
  - demand d_i
  - service time s_i
  - time window $[a_i, b_i]$
- For each arc (i, j)
  - cost c_{ij}
  - travel time t_{ij}
- $d_0 = d_{n+1} = s_0 = s_{n+1} = 0$
- $[a_0, b_0] = [a_{n+1}, b_{n+1}]$ planning horizon
- Each vehicle has max capacity Q
- Each feasible route must satisfy capacity, elementarity, and time window constraints

The VRPTW with Waiting Time Costs

- In the VRPTW, the classical cost is total distance traveled, or total travel time.
- In the VRPTWWTC, the objective is to minimize the total route duration, which includes total travel time, service time and *total waiting time* (e.g., cost of driver per time unit).
- Route start time $T_0 \in]-\infty, \infty[$ is a decision variable.
- Maximum duration defined for each route (e.g., maximum driver shift).

Formulation

Set Covering Model

$$\min \sum_{r \in \Omega} c_r y_r$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} y_r \geq 1 \qquad \forall i \in N,$$

$$y_r \in \{0,1\} \qquad \forall r \in \Omega.$$

- Ω is the set of feasible routes
- $a_{ir} = 1$ if and only if route r visits i, 0 otherwise
- $y_r = 1$ if and only if route r is part of the solution, 0 otherwise
- c_r cost of route r

Table of Contents

Branch-and-Price Overview

- Branch-and-Price (BP): Branch-and-bound + Column Generation (CG)
- CG: Restricted Master Problem $\leftrightarrows$ Pricing Problem
- Master Problem: Set Covering

Restricted Master Problem

$$\min \sum_{r \in \overline{\Omega}} c_r y_r$$

$$\text{s.t.} \sum_{r \in \overline{\Omega}} a_{ir} y_r \geq 1 \qquad \forall i \in N,$$

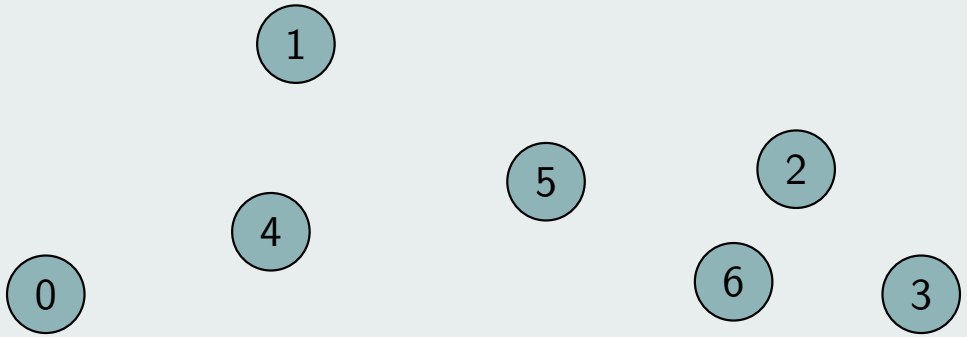
$$y_r \geq 0 \qquad \forall r \in \overline{\Omega} \subset \Omega.$$

- Pricing Problem: Elementary Shortest Path Problem with Resource Constraints (ESPPRC)
- ESPPRC is NP-hard![1]

[1] Dror 1994.

Labeling algorithm[2]

Label extension



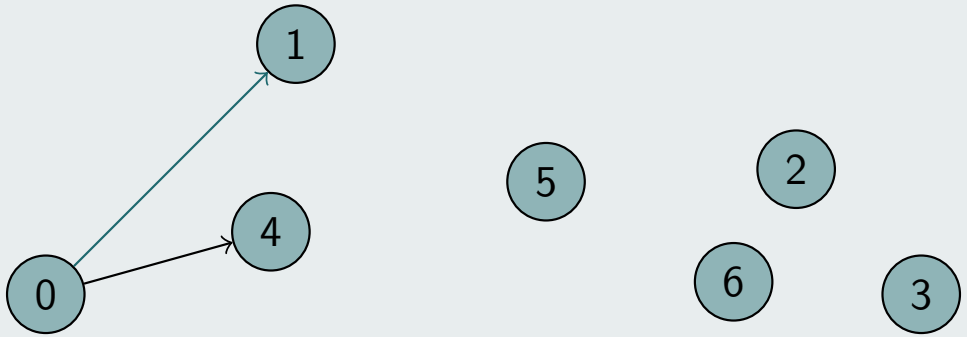
$$L_0 = (C_0 = 0, \mathbf{R}_0 = \mathbf{0}, \{E_0^k\}_{k \in N} = \mathbf{0})$$

- Each label L_i represents a partial path ending at node i.

[2] Feillet et al. 2004.

Labeling algorithm[2]

Label extension



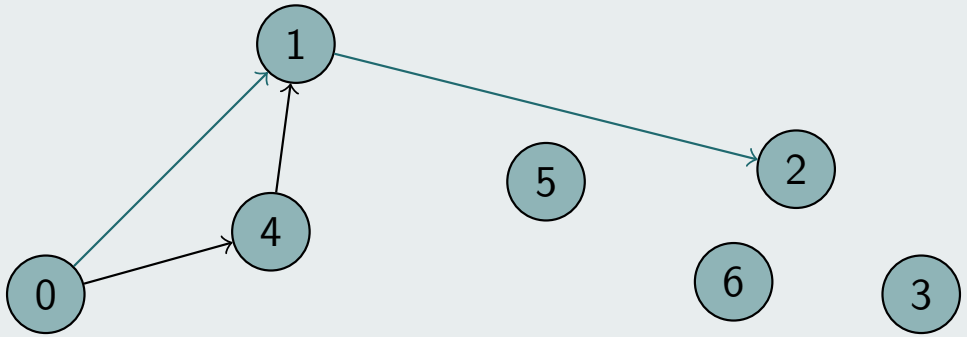
$$L_1 = (C_1, \mathbf{R}_1, \{E_1^k\}_{k \in N})$$

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.

[2] Feillet et al. 2004.

Labeling algorithm[2]

Label extension



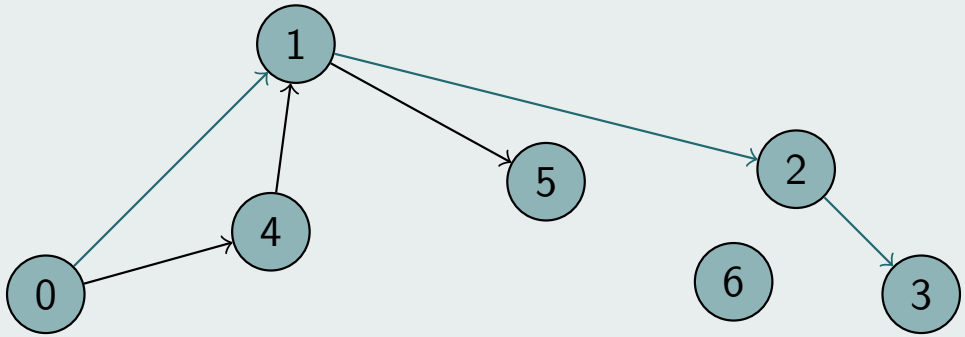
$$L_2 = (C_2, \mathbf{R}_2, \{E_2^k\}_{k \in N})$$

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.
- Complexity depends on number of maintained labels.

[2] Feillet et al. 2004.

Labeling algorithm[2]

Label extension



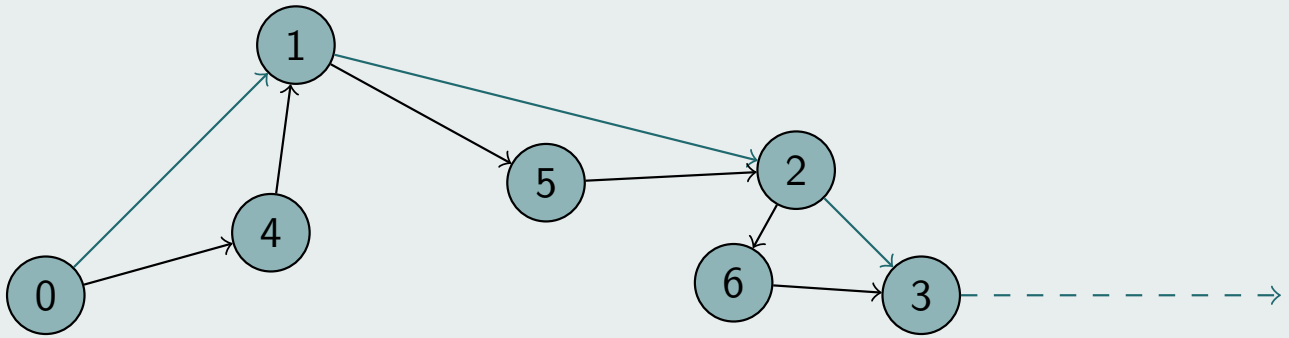
$$L_3 = (C_3, \mathbf{R}_3, \{E_3^k\}_{k \in N})$$

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.
- Complexity depends on number of maintained labels.
- Resources are used for domination and feasibility checks.

[2] Feillet et al. 2004.

Labeling algorithm[2]

Label extension



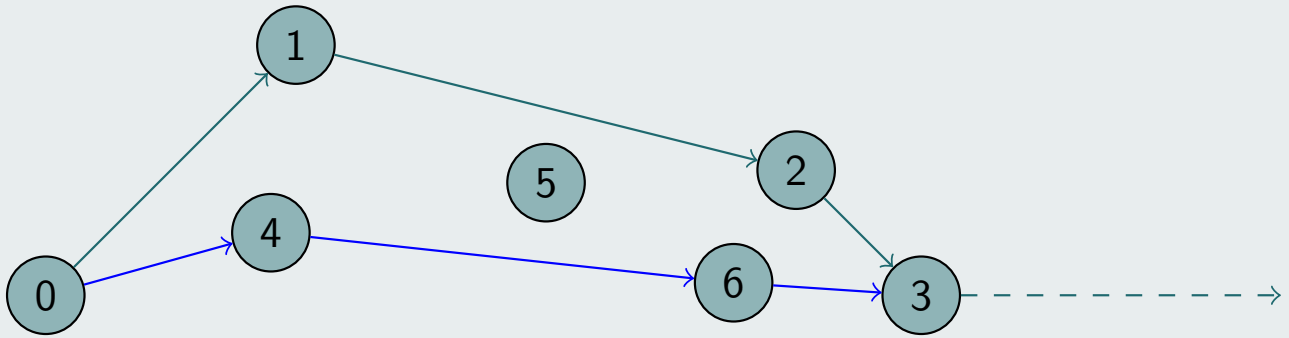
$$L_3 = (C_3, \mathbf{R}_3, E_3^1, E_3^2, \ldots, E_3^n)$$
$$|\wedge$$
$$L_3' = (C_3', \mathbf{R}_3', E_3'^1, E_3'^2, \ldots, E_3'^n)$$

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.
- Complexity depends on number of maintained labels.
- Resources are used for domination and feasibility checks.

[2] Feillet et al. 2004.

Labeling algorithm[2]

Label extension



$$L_3 = (C_3, \mathbf{R}_3, E_3^1, E_3^2, \ldots, E_3^n)$$

$$L_3'' = (C_3'', \mathbf{R}_3'', E_3''^1, E_3''^2, \ldots, E_3''^n)$$

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.
- Complexity depends on number of maintained labels.
- Resources are used for domination and feasibility checks.
- Binary resources $\{E_i^k\}_{k \in N}$ ensure elementarity, but make domination harder.

[2] Feillet et al. 2004.

Adapting the labeling algorithm to the VRPTWWTC

VRPTW label structure

$$L_i = (C_i, \tau_i, q_i, \{E_i^k\}_{k \in N})$$

- C_i accumulated reduced cost
- τ_i total duration
- q_i accumulated vehicle load
- $\{E_i^k\}_{k \in N}$ elementarity resources

VRPTWWTC label structure

$$\lambda_i = (\delta_i, \nu_i, \mu_i, \zeta_i, q_i, \{E_i^k\}_{k \in N})$$

- δ_i sum of dual prices
- ν_i latest feasible start time from the depot
- μ_i earliest feasible service time at i
- ζ_i minimum possible duration of the partial path

Label Domination Rules

VRPTW

$$L_i \leq L'_i \Leftrightarrow \begin{cases} C_i \leq C'_i \\ \tau_i \leq \tau'_i \\ q_i \leq q'_i \\ E_i^k \leq {E'}_i^k, \ \forall k \in N \end{cases}$$

VRPTWWTC

$$\lambda_i \leq \lambda'_i \Leftrightarrow \begin{cases} \delta_i + \min\{\nu_i - \nu'_i, 0\} \geq \delta'_i \\ \mu_i \leq \mu'_i \\ \zeta'_i \leq \zeta'_i \\ q_i \leq q'_i \\ E_i^k \leq {E'}_i^k, \ \forall k \in N \end{cases}$$

Additional BP Elements

- Bidirectional bounded labeling algorithm
  - Extend labels from source and sink
  - Stop extension when a critical resource is halfway consumed
  - Concatenate forward and backward labels, take care of duplicates
- Labeling algorithm allows insertion of multiple columns in RMP
- Branching on fractional arc flow
  - Problems at child nodes are more balanced

Table of Contents

Decremental State Space Relaxation (DSSR)[3]

- Maintain a set Θ of *critical* nodes on which elementarity is enforced.

DSSR

DSSR

1: Initialize Θ
2: **repeat**
3: $\quad P \leftarrow \text{LabelExtension}(\Theta)$
4: $\quad \Phi \leftarrow \text{MultipleVisits}(P)$
5: $\quad \Theta \leftarrow \Theta \cup \Phi$
6: **until** P is elementary

- We manipulate the state space in a *global* way.

[3] Boland, Dethridge, and Dumitrescu 2006; Righini and Salani 2008.

ng-route Relaxation (NGRR)[4]

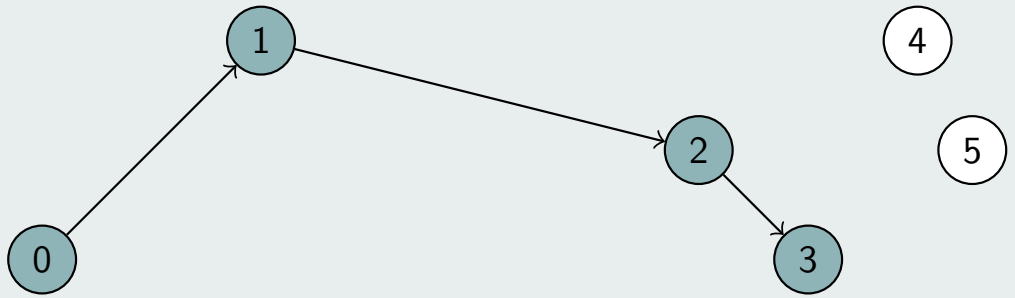
NGRR extension

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



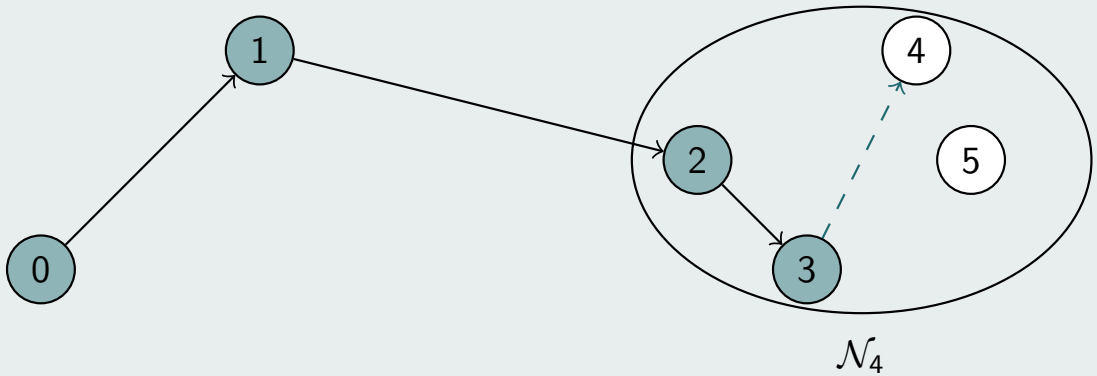
- $\mathbf{E}_3 = \{0, 1, 2, 3\}$

- Define neighbourhoods $\mathcal{N}_i$, $\forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



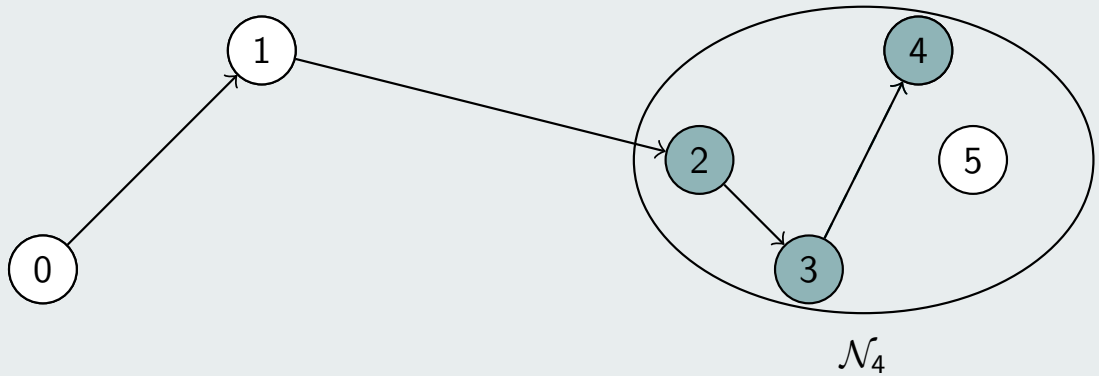
- $\mathbf{E}_3 = \{0, 1, 2, 3\}$
- $j = 4,\ \mathcal{N}_4 = \{2, 3, 4, 5\}$

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in \mathbf{E}_i$, insert k in $\mathbf{E}_j$ only if $k \in \mathcal{N}_j$.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



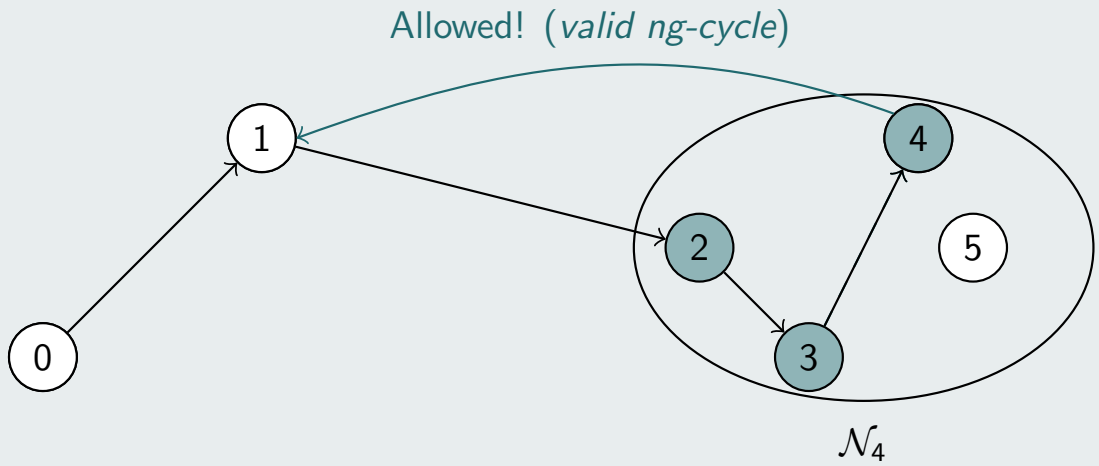
- $\mathbf{E}_3 = \{0, 1, 2, 3\}$
- $j = 4,\ \mathcal{N}_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap \mathcal{N}_4) \cup \{4\} = \{2, 3, 4\}$

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in \mathbf{E}_i$, insert k in $\mathbf{E}_j$ only if $k \in \mathcal{N}_j$.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



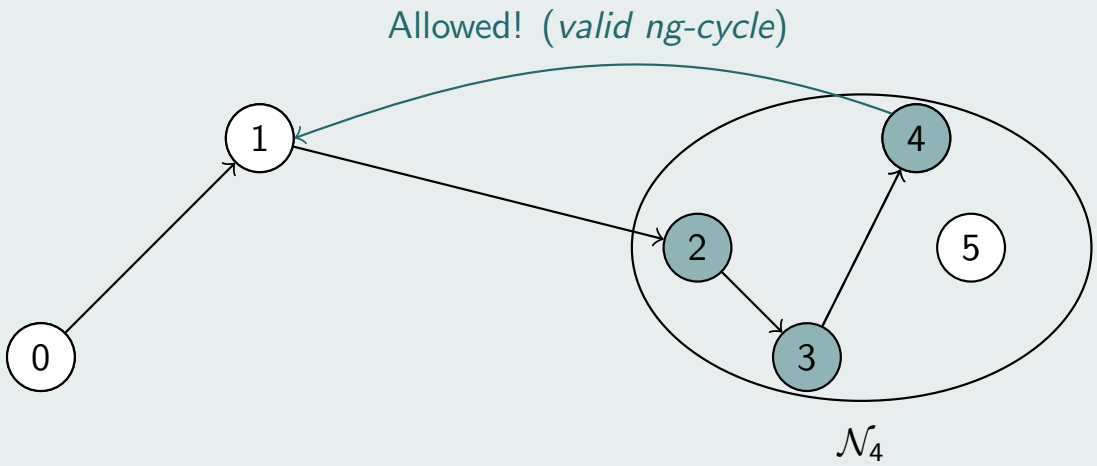
- $\mathbf{E}_3 = \{0, 1, 2, 3\}$
- $j = 4,\ \mathcal{N}_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap \mathcal{N}_4) \cup \{4\} = \{2, 3, 4\}$

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in \mathbf{E}_i$, insert k in $\mathbf{E}_j$ only if $k \in \mathcal{N}_j$.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



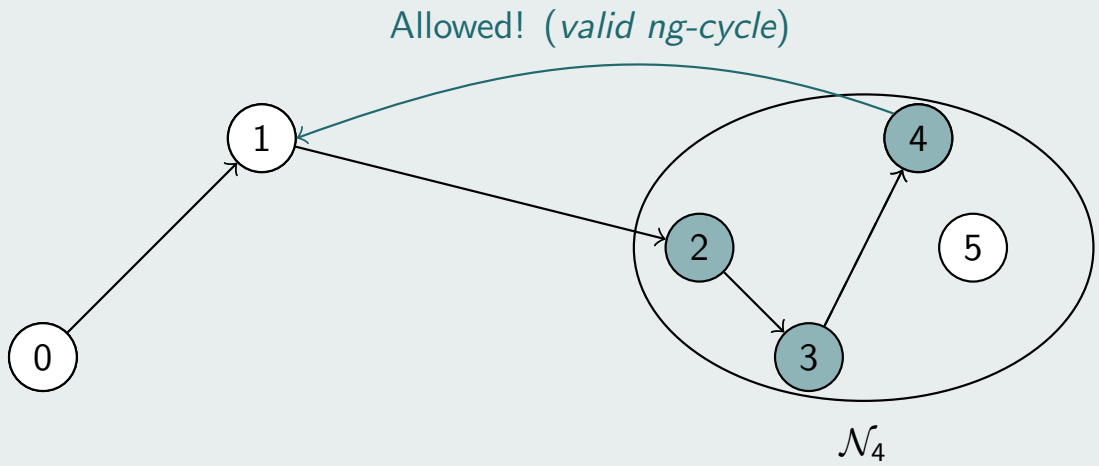
- $\mathbf{E}_3 = \{0, 1, 2, 3\}$
- $j = 4,\ \mathcal{N}_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap \mathcal{N}_4) \cup \{4\} = \{2, 3, 4\}$

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in \mathbf{E}_i$, insert k in $\mathbf{E}_j$ only if $k \in \mathcal{N}_j$.
- Resulting path may **not** be elementary.

[4] Baldacci et al. 2010.

ng-route Relaxation (NGRR)[4]

NGRR extension



- $\mathbf{E}_3 = \{0, 1, 2, 3\}$
- $j = 4,\ \mathcal{N}_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap \mathcal{N}_4) \cup \{4\} = \{2, 3, 4\}$

- Define neighbourhoods $\mathcal{N}_i,\ \forall i \in N$.
- $\mathbf{E}_i$ = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in \mathbf{E}_i$, insert k in $\mathbf{E}_j$ only if $k \in \mathcal{N}_j$.
- Resulting path may **not** be elementary.
- We manipulate the state space in a *local* way.

[4] Baldacci et al. 2010.

NGRR and DSSR: Global NG-DSSR

- We can combine NGRR and DSSR in several ways.
- Several hybrid algorithms have been considered in the literature.
- This straightforward hybridization uses both neighbourhoods and the set of critical nodes[5].

Global NG-DSSR

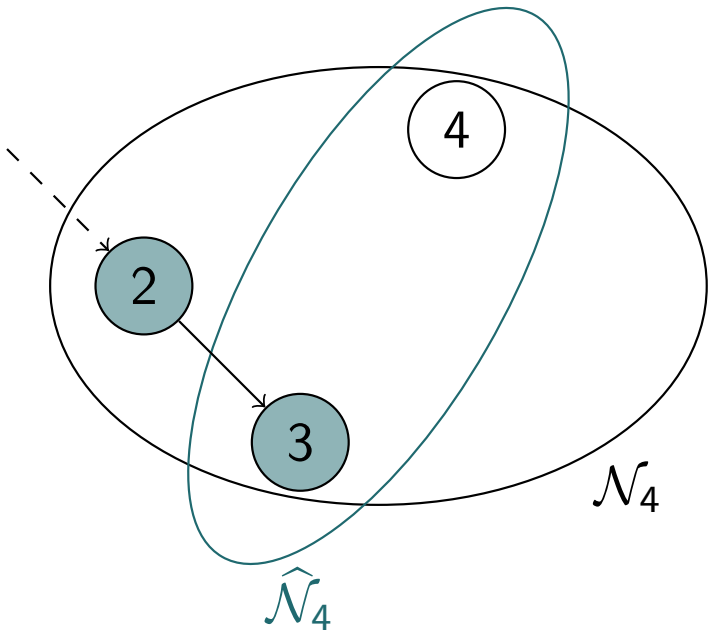
NG-DSSR-G

1: Initialize Θ, $M_i \ \forall i \in N$
2: **repeat**
3: $\quad P \leftarrow \textsc{LabelExtension}(\Theta, \mathcal{N}_i)$
4: $\quad \Phi \leftarrow \textsc{MultipleVisits}(P, \{\mathcal{N}_i\}_{i \in N})$ // Only nodes in invalid ng-cycles
5: $\quad \Theta \leftarrow \Theta \cup \Phi$
6: **until** P is ng-route

[5] Dayarian et al. 2015.

NGRR and DSSR: Local NG-DSSR

- Define *applied* neighbourhoods $\widehat{\mathcal{N}}_i \subseteq \mathcal{N}_i,\ \forall i \in N$ to use throughout label extension[6].
- When best path has invalid cycle C, update applied neighbourhoods of nodes in C.



Local NG-DSSR

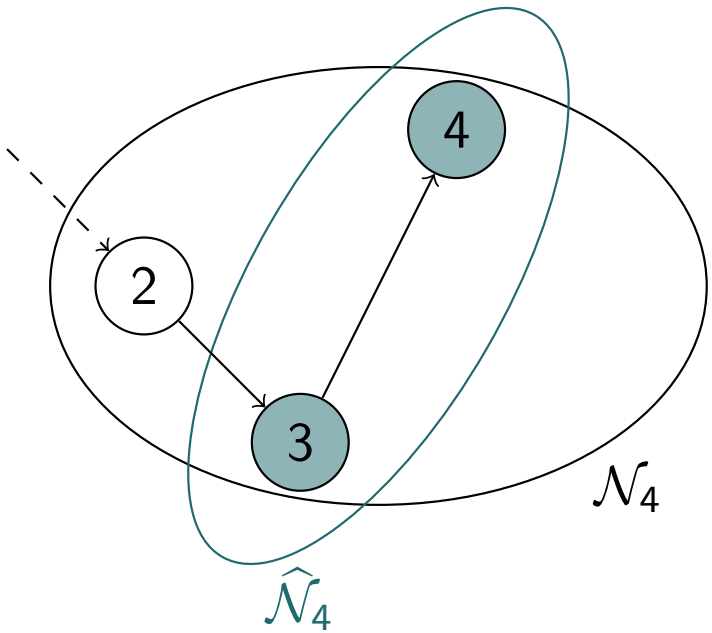
NG-DSSR-L

1: Initialize $M_i,\ \widehat{\mathcal{N}}_i,\ \forall i \in N$
2: **repeat**
3: $\quad P \leftarrow \textsc{LabelExtension}(\{\widehat{\mathcal{N}}_i\}_{i \in N})$
4: $\quad C \leftarrow \textsc{InvalidCycle}(P, \{\mathcal{N}_i\}_{i \in N})$
5: $\quad \textsc{Update}(\{\widehat{\mathcal{N}}_i\}_{i \in N}, C)$
6: **until** P is ng-route

[6] Dayarian et al. 2015; Martinelli, Pecin, and Poggi 2014.

NGRR and DSSR: Local NG-DSSR

- Define *applied* neighbourhoods $\widehat{\mathcal{N}}_i \subseteq \mathcal{N}_i,\ \forall i \in N$ to use throughout label extension[6].
- When best path has invalid cycle C, update applied neighbourhoods of nodes in C.



Local NG-DSSR

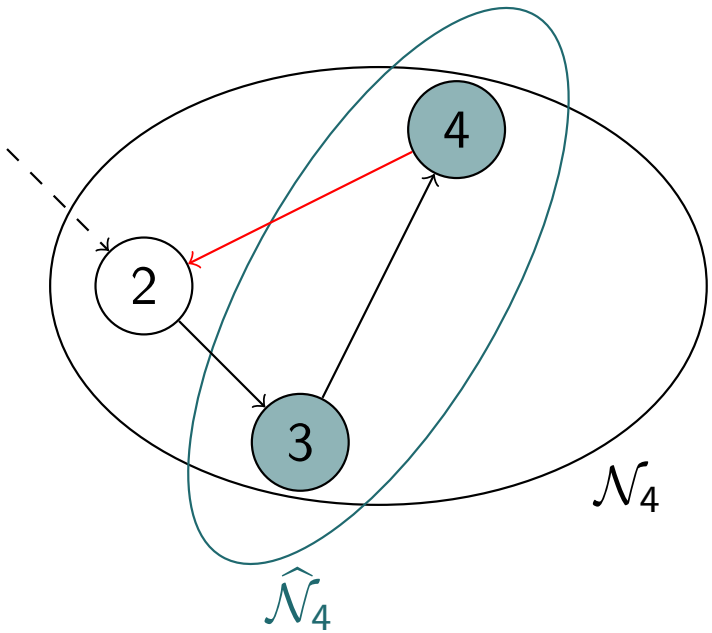
NG-DSSR-L

1: Initialize $M_i,\ \widehat{\mathcal{N}}_i,\ \forall i \in N$
2: **repeat**
3: $\quad P \leftarrow \textsc{LabelExtension}(\{\widehat{\mathcal{N}}_i\}_{i \in N})$
4: $\quad C \leftarrow \textsc{InvalidCycle}(P, \{\mathcal{N}_i\}_{i \in N})$
5: $\quad \textsc{Update}(\{\widehat{\mathcal{N}}_i\}_{i \in N}, C)$
6: **until** P is ng-route

[6] Dayarian et al. 2015; Martinelli, Pecin, and Poggi 2014.

NGRR and DSSR: Local NG-DSSR

- Define *applied* neighbourhoods $\widehat{\mathcal{N}}_i \subseteq \mathcal{N}_i,\ \forall i \in N$ to use throughout label extension[6].
- When best path has invalid cycle C, update applied neighbourhoods of nodes in C.



Local NG-DSSR

NG-DSSR-L

1: Initialize $M_i,\ \widehat{\mathcal{N}}_i,\ \forall i \in N$
2: **repeat**
3: $\quad P \leftarrow \textsc{LabelExtension}(\{\widehat{\mathcal{N}}_i\}_{i \in N})$
4: $\quad C \leftarrow \textsc{InvalidCycle}(P, \{\mathcal{N}_i\}_{i \in N})$
5: $\quad \textsc{Update}(\{\widehat{\mathcal{N}}_i\}_{i \in N}, C)$
6: **until** P is ng-route

[6] Dayarian et al. 2015; Martinelli, Pecin, and Poggi 2014.

NGRR and DSSR

- We can derive a different version of DSSR using this local approach[7].
- Define local critical sets $\widehat{\Theta}_i,\ \forall i \in N$.
- When best path has cycle C, update critical sets of nodes in C.

Local DSSR

DSSR-L

1: Initialize $\widehat{\Theta}_i,\ \forall i \in N$
2: **repeat**
3: $\quad P \leftarrow \textsc{LabelExtension}(\{\widehat{\Theta}_i\}_i)$
4: $\quad C \leftarrow \textsc{Cycle}(P)$
5: $\quad \textsc{Update}(\{\widehat{\Theta}_i\}_i, C)$
6: **until** P is elementary

[7] Martinelli, Pecin, and Poggi 2014.

NGRR and DSSR

- Global and Local NG-DSSR output ng-routes.
- We can continue execution with DSSR (DSSR-L) to obtain elementary routes.

Elementary Global NG-DSSR

NG-DSSR-G-E

1: $P \leftarrow$ NG-DSSR-G()
2: **if** P is not elementary **then**
3: $P \leftarrow$DSSR(P)

Elementary Local NG-DSSR

NG-DSSR-L-E

1: $P \leftarrow$NG-DSSR-L()
2: **if** P is not elementary **then**
3: $P \leftarrow$DSSR-L(P)

Labeling Algorithms Overview

ng-route algorithms	elementary route algorithms
NGRR	DSSR
NG-DSSR-G	DSSR-L
NG-DSSR-L	NG-DSSR-G-E
	NG-DSSR-L-E

How to rigorously compare these algorithms?

Table of Contents

Parameters — DSSR

- Critical set initialization:
  - `dssr_init_s`: *none*, *tca*, *hca*, *wtca*, *whca*, *mixed*
  - `dssr_init_n`: $]0, 1[$
- Critical set update rules:
  - `dssr_path_s`: *all-paths*, *one-path*, *in-between*
  - `dssr_path_n`: $]0, 1[$
  - `dssr_node_s`: *all-nodes*, *one-node*, *in-between*
  - `dssr_node_n`: $]0, 1[$

Parameters — NGRR

- Neighborhood size `ng_size`: $]0, 1[$
- Node metric `ng_type`: *travel-time*, *cycle-risk*, *mixed*
- Coefficient for the mixed metric `ng_mix`: $]0, 1[$

Branch-and-Price Parameters

- Maximum number of generated paths `concat_stop`: $]0, 10000]$
- Maximum number of paths inserted in the master problem `n_col`: $]0, 1[$
- Tree navigation strategy `tree_nav`: depth-first search, breadth-first search, or best-first search

Parameters

Parameter	DSSR	NGRR	NG-D.-G	NG-D.-L	DSSR-L	NG-D.-G-E	NG-D.-L-E	Range
`tree_trav`	×	×	×	×	×	×	×	{`breadth`, `depth`, `best`}
`n_conc`	×	×	×	×	×	×	×	$]0, 10000]$
`n_col`	×	×	×	×	×	×	×	$]0, 1]$
`dssr_init_s`	×		×		×	×		{`hca`, `tca`, `whca`, `wtca`, `mix`}
`dssr_init_n`	×		×		×	×		$]0, 1[$
`dssr_path_s`	×		×	×	×	×	×	{`1_p`, `in_btw`, `all_p`}
`dssr_path_n`	×		×	×	×	×	×	$]0, 1[$
`dssr_node_s`	×		×			×		{`1_n`, `in_btw`, `all_n`}
`dssr_node_n`	×		×			×		$]0, 1[$
`ng_type`		×	×	×		×	×	{`tt`, `ccr`, `mix`}
`ng_size`		×	×	×		×	×	$]0, 1[$
`ng_mix`		×	×	×		×	×	$]0, 1[$

Table: Parameters choices

- All parameters except `tree_trav` have a duplicate that controls the same value at the root node of the BB tree

Table of Contents

Methodology

Tuning:

- Tune the parameters with the `irace` package[8]
- `irace` takes a configurable algorithms and a training instance set and returns a set of parameter configurations
- Tuning is performed for each algorithm and for each problem
- To avoid overtuning, the tuning set is derived from the Gehring & Homberger instances for the VRPTW instead of the Solomon instances
- Set of 6 instances, one per type (clustered, random, random-clustered) and per size (25 and 50 customers)
- Tuning budget of 5000 experiments for each algorithm, time limit $TL = 3$ hours per experiment

[8] López-Ibáñez et al. 2016.

Methodology

Measure of performance for configuration θ on instance i

$$t'(i,\theta) = \begin{cases} t(i,\theta) & \text{if } t(i,\theta) < TL \text{ seconds (3 hours)}, \\ TL + UB(i,\theta) & \text{if root node solved and } TL \text{ reached}, \\ M & \text{otherwise} \end{cases}$$

Benchmarking:

- Run the best configuration of each algorithm on the Solomon instances
- Perform Friedman test + post-hoc Wilcoxon signed rank test on the results

Statistical tests on Solomon instances — First Round

VRPTW

- Friedman's test reports $p < 2.2 \times 10^{-16}$, denoting statistically significant difference among the data
- Post-hoc test reports that NG-DSSR-L and NG-DSSR-L-E outperform DSSR, NG-DSSR-G and NG-DSSR-G-E with $p < 0.01$
- No significant difference between NG-DSSR-L, NG-DSSR-L-E, DSSR-L and NGRR

VRPTWWTC

- Friedman's test reports again $p < 2.2 \times 10^{-16}$
- Post-hoc test reports that DSSR-L outperforms DSSR, NGRR, NG-DSSR-G, and NG-DSSR-G-E with a $p < 1.0 \times 10^{-7}$
- DSSR-L, NG-DSSR-L and NG-DSSR-L-E are statistically equivalent

Table: VRPTW — Elementary algorithms — 3 hour time limit.

		DSSR				DSSR-L				NG-DSSR-G-E				NG-DSSR-L-E			
Class	NI	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)
C1–25	9	9		10.23		9		5.27		9		140.3		9		10.03	
R1–25	12	12		1.20		12		1.23		12		1.87		12		1.20	
RC1–25	8	8		5.26		8		3.80		8		18.17		8		4.46	
C1–50	9	8		13.84		9		105.4		8		82.46		9		184.9	
R1–50	12	11	1	435.8	4.59	11	1	193.8	4.47	11	1	1015	4.59	11	1	154.5	3.99
RC1–50	8	2	6	1863	12.01	2	6	1682	12.64	2	6	3442	12.11	2	6	1723	11.0

Table: VRPTW — *ng*-route algorithms — 3 hour time limit.

		NGRR				NG-DSSR-G				NG-DSSR-L			
Class	NI	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)
C1–25	9	9		1.49		9		3.22		9		2.25	
R1–25	12	12		0.95		12		2.02		12		1.14	
RC1–25	8	8		3.22		8		3.49		8		3.91	
C1–50	9	9		40.39		9		491.0		9		55.56	
R1–50	12	11	1	495.1	5.22	10	2	65.67	3.75	11	1	127.6	4.34
RC1–50	8	2	6	2478	12.27	2	6	3963	10.63	2	6	1683	11.8

Table: VRPTWWTC — Elementary algorithms — 3 hour time limit.

		DSSR				DSSR-L				NG-DSSR-G-E				NG-DSSR-L-E			
Class	NI	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)
C1–25	9	5	1	1030	0.28	6	3	874.1	0.20	5	1	1016	0.33	5	3	75.1	0.19
R1–25	12	12		103.7		12		11.7		12		145.7		12		23.2	
RC1–25	8	8		59.4		8		7.91		8		56.5		8		18.8	
C1–50	9	4		167		6		907.2		5		2065		6	1	923.7	0.01
R1–50	12	5	6	2211	1.17	9	3	1342	1.99	5	7	1102	1.24	10	2	2458	1.73
RC1–50	8		6		6.83	1	7	209.7	6.43	1	7	7800	6.63	1	7	1487	6.39

Table: VRPTWWTC — *ng*-route algorithms — 3 hour time limit.

		NGRR				NG-DSSR-G				NG-DSSR-L			
Class	NI	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)	S	PS	Time	Gap (%)
C1–25	9	5	4	238.7	0.27	5	2	1142	0.31	5	4	109.5	0.41
R1–25	12	12		37.9		12		74.8		12		12.9	
RC1–25	8	8		1643		8		158.8		8		29.2	
C1–50	9	6		1754		4		180.0		5	1	211.8	0.02
R1–50	12	7	5	1362	2.12	6	6	2519	1.63	9	3	1893	2.08
RC1–50	8	1	7	9347	6.17	1	7	2175	6.53	1	7	359.7	6.66

Benchmarking — Second Round

- Consider the best four algorithms (DSSR-L, NG-DSSR-L, NG-DSSR-L-E, NGRR), with the same configurations used previously
- We run them on the Solomon instances for 6 hours, on 25, 50, 75 and 100 customers

VRPTW

- Friedman's test reports no significant difference ($p = 0.11$)

VRPTWWTC

- The Friedman test reports a p-value $p = 5.093 \times 10^{-5}$
- Post-hoc reports that NGRR is rejected with p-value $p < 0.01$ by the other algorithms

Benchmarking — Second Round

Table: VRPTW — Benchmarks — 6 hours.

		DSSR-L					NG-DSSR-L-E					NGRR					NG-DSSR-L				
Class	NI	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)
C1–50	9	9			105.4		9			184.9		9			40.39		9			55.56	
R1–50	12	11	1		193.8	4.04	11	1		154.5	3.87	11	1		495.1	4.95	11	1		127.6	4.07
RC1–50	8	2	6		1682	12.22	2	6		1723	10.97	2	6		2478	12.01	2	6		1683	11.12
C1–75	9	8	1		835.7	3.05	8	1		610.6	3.45	8	1		1545	3.39	8	1		475.4	3.39
R1–75	12	10	2		3679	2.38	10	2		2079	2.48	9	3		2447	2.95	10	2		3633	2.90
RC1–75	8	1	5	2	924.5	6.36	2	6		9868	6.79	1	6	0	580.7	6.47	1	7		985.0	6.20
C1–100	9	9			358.7		9			288.0		9			2063		9			296.5	
R1–100	12	6	4	2	4738	2.01	6	5	1	4108	2.40	4	8		353.3	2.11	6	5	0	4096	1.95
RC1–100	8		5	3		3.96		5	3		4.97		5	2		6.16		6	1		4.84

Benchmarking — Second Round

Table: VRPTWWTC — Benchmarks — 6 hours.

		DSSR-L					NG-DSSR-L-E					NGRR					NG-DSSR-L				
Class	NI	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)	S	PS	R	Time	Gap (%)
C1–25	9	7	2		3533	0.32	6	3		3054	0.21	6	3		3465	0.27	7	2		5201	1.49
R1–25	12	12			13.2		12			23.1		12			33.3		12			13.1	
RC1–25	8	8			11.2		8			24.3		8			1968		8			96.8	
C1–50	9	6	1	1	1701	0.01	6	1	1	607.2	0.01	6		2	930.9		6	1	2	926.8	0.02
R1–50	12	9	3		1528	2.03	10	2		3601	2.06	8	4		1676	2.16	9	3		1898	2.02
RC1–50	8	2	6		10478	6.72	1	7		1213	6.39	1	7		8058	7.21	1	7		737.8	6.70
C1–75	9	5	1	3	4274	0.21	5	1	2	1787	0.20	5		4	3614		5		3	2992	
R1–75	12	3	9		54.3	1.35	4	8		4923	1.34	4	8		1986	1.43	4	8		2925	1.47
RC1–75	8		8			3.12		8			3.03		8			3.50	1	7		17613	3.45
C1–100	9	5		3	1896		5		3	955.4		5		3	4172		5		4	1702	
R1–100	12	2	2	5	9121	0.74	3	1	6	10726	0.80	2	2	3	7270	0.40	3	1	5	6492	0.80
RC1–100	8		2	6		1.27		2	6		1.72		3	5		1.97		5	2		2.05

Parameter Configurations Overview — VRPTWWTC

	n_col_root	n_conc_root	n_col	n_conc	tree_nav
DSSR-L					
Best	82%	2820	31%	9247	best-first
Avg	42%	3998	35%	8902	best-first (5)
Min	10%	2764	22%	7308	
Max	81%	8481	63%	9867	
NG-DSSR-L-E					
Best	27%	7910	42%	9845	depth-first
Avg	45%	5949	37%	8688	depth-first (6)
Min	26%	1003	36%	5234	
Max	98%	8320	42%	9923	
NG-DSSR-L					
Best	4%	8752	14%	3079	depth-first
Avg	10%	6900	24%	3540	depth-first (6)
Min	1%	5476	11%	1100	
Max	35%	8752	63%	4519	

Table: BP parameters

Parameter Configurations Overview — VRPTWWTC

	dssr_init_s_rt	dssr_init_n_rt	dssr_path_s_rt	dssr_path_n_rt
DSSR-L				
Best	none	n/a	in-between	25%
Avg	none (6)		in-between (6)	38%
Min				24%
Max				47%
NG-DSSR-L-E				
Best	n/a	n/a	one-path	n/a
Avg			one-path (6)	
NG-DSSR-L				
Best	n/a	n/a	all-paths	n/a
Avg			all-paths (5)	

Table: DSSR parameters, root

Parameter Configurations Overview — VRPTWWTC

	dssr_init_s	dssr_init_n	dssr_path_s	dssr_path_n
DSSR-L				
Best	none	n/a	in-between	39%
Avg	none (6)		in-between (6)	57%
Min				38%
Max				80%
NG-DSSR-L-E				
Best	n/a	n/a	in-between	98%
Avg			in-between (6)	81%
Min				72%
Max				97%
NG-DSSR-L				
Best	n/a	n/a	in-between	35%
Avg			in-between (6)	31%
Min				26%
Max				34%

Table: DSSR parameters

Parameter configurations Overview — VRPTWWTC

	ng_m_root	ng_s_root	ng_m	ng_s
NG-DSSR-L-E				
Best	travel-time	6%	travel-time	23%
Avg	travel-time (5)	9%	travel-time (6)	24%
Min		4%		16%
Max		21%		45%
NG-DSSR-L				
Best	travel-time	17%	travel-time	88%
Avg	travel-time (6)	17%	travel-time (5)	81%
Min		14%		50%
Max		24%		97%

Table: NG parameters

Table of Contents

Conclusions

- Important to rigorously compare performance-critical routines
- "Local" approach for treating elementarity resources seems the best approach
- It makes sense to parametrize differently the algorithm in the root node
- Best parameter values provided by tuning might be different than the ones suggested in the literature
- Possible future work:
  - Integrate in state-of-the-art branch-and-cut-and-price
  - Develop into matheuristic
  - Adapt to multi-trip extension of the problem

Thanks!

Thanks for your attention.