A Bucket Graph Based Labelling Algorithm for Vehicle Routing Pricing

Ruslan Sadykov[1,2] Artur Pessoa[3] Eduardo Uchoa[3]

[1] Inria Bordeaux, France

[2] Université Bordeaux, France

[3] Universidade Federal Fluminense, Brazil

POC Autumn School on Advanced BCP Tools
Paris, November 22, 2019

Contents

Resource-constrained (elementary) shortest path problem, or RC(E)SPP

- A directed graph $G = (V, A)$, a source and a sink.
- Set R of resources
- For each arc $a \in A$
  - cost c_a
  - resource consumption $q_{a,r}$, $r \in R$
  - accumulated resource consumption bounds $[l_{a,r}, u_{a,r}]$, $r \in R$

Objective

Find an (elementary) path from the source to the sink which minimizes the total cost.

Literature : “standalone” algorithms for the RC(E)SPP

Test instances with a sparse graph (often acyclic) with few global resources, aim to find one optimal solution

- Heavy pre-processing and Lagrangian relaxation [Dumitrescu and Boland, 2003]
- Transformation to the shortest path problem [Zhu and Wilhelm, 2012]
- Transformation the k-shortest paths problem [Santos et al., 2007] [Sedeno-Noda and Alonso-Rodríguez, 2015]
- Pulse Algorithm (depth-first search, pruning by limited dominance and bounds) [Lozano and Medaglia, 2013]
- Bi-directional A^* [Thomas et al., 2019]
- Best performance is by [Lozano and Medaglia, 2013] [Sedeno-Noda and Alonso-Rodríguez, 2015] [Thomas et al., 2019]

Labeling algorithm

- Every label represents a partial path starting from the source.
- Label L contains
  - v^L — last visited vertex
  - c^L — current total cost
  - q^L — current accumulated resource consumption
  - $\mathcal{V}^L$ — set of visited vertices

Dominance

Label L dominates L' if any feasible completion of L' is feasible for L and has larger or the same cost.

Sufficient condition: label L dominates L' if

$$v^L = v^{L'}, \quad c^L \leq c^{L'}, \quad q^L \leq q^{L'}, \quad \mathcal{V}^L \subseteq \mathcal{V}^{L'}.$$

Basic labelling algorithm

$\mathcal{L} = \bigcup_{v \in V} \mathcal{L}_v$ — set of non-extended labels
$\mathcal{E} = \bigcup_{v \in V} \mathcal{E}_v$ — set of extended labels

---

$\mathcal{L} \to \{(\text{source}, 0, 0, 0, \{\text{source}\})\}$, $\mathcal{E} \leftarrow \emptyset$
while $\mathcal{L} \neq \emptyset$ **do**
  pick a label L in $\mathcal{L}$, $v^L \neq \text{sink}$
  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{L\}$, $\mathcal{E} \leftarrow \mathcal{E} \cup \{L\}$
  foreach $v \in V \setminus v^L$ **do**
    extend L to L' along arc (v^L, v)
    if L' *is feasible and not dominated by a label in* $\mathcal{L}_v \cup \mathcal{E}_v$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{L'\}$
      remove from $\mathcal{L}_v \cup \mathcal{E}_v$ all labels dominated by L'
  return a label in $\mathcal{L}_{\text{sink}}$ with the smallest reduced cost

---

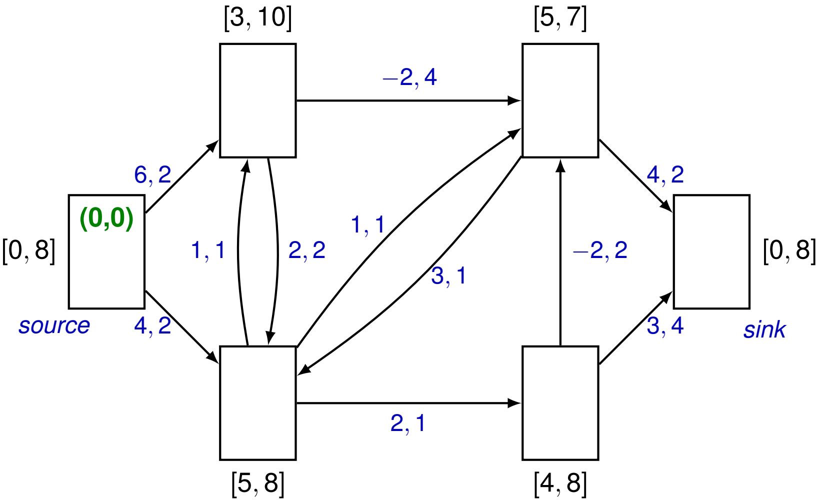
Label-setting if labels are picked in a total order $\leq_{\text{lex}}$ such that

$$L \text{ extends to } L' \Rightarrow L \leq_{\text{lex}} L', \quad L \text{ dominates } L' \Rightarrow L \leq_{\text{lex}} L'$$

Otherwise, it is label-correcting (for example, cycling over $\mathcal{L}_v$)

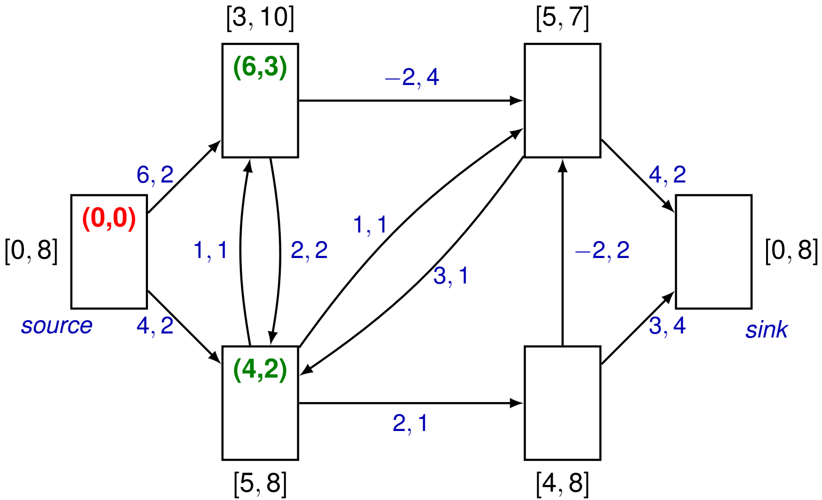
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



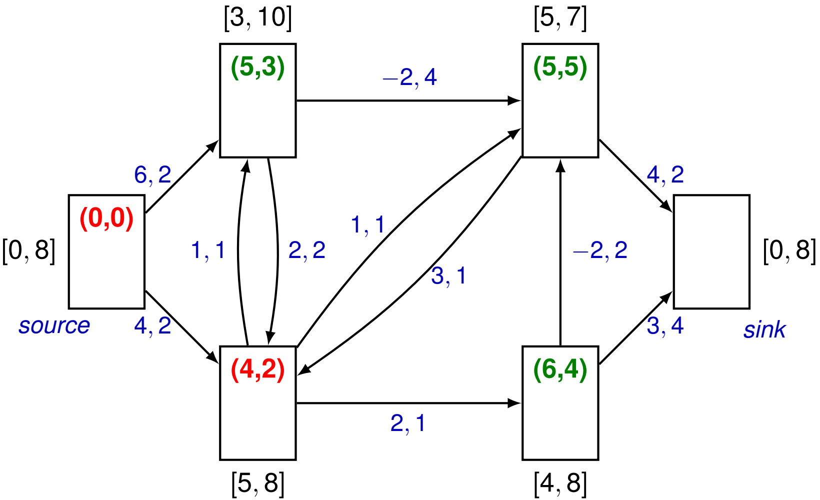
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



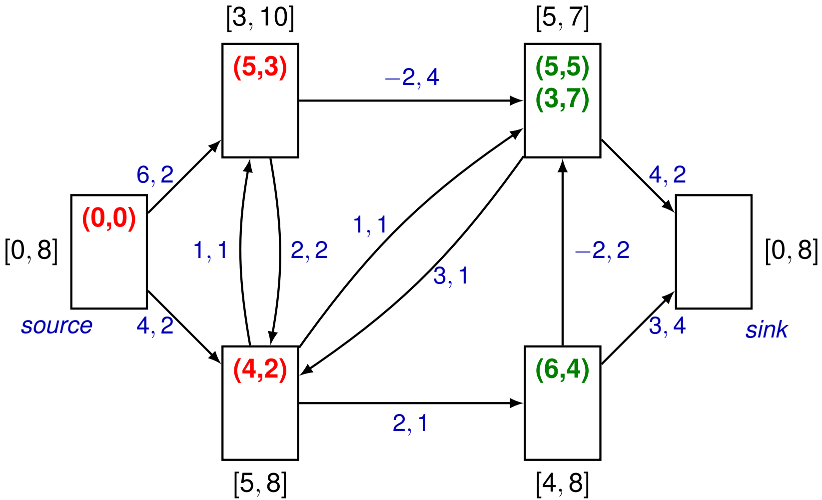
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



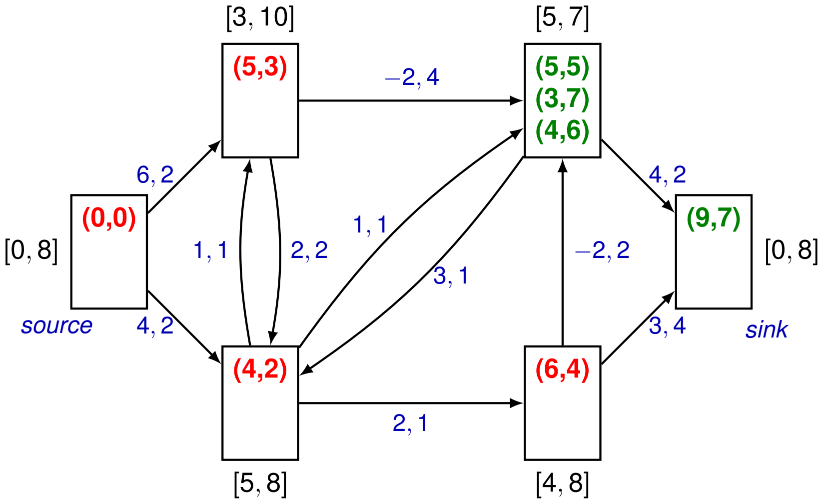
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



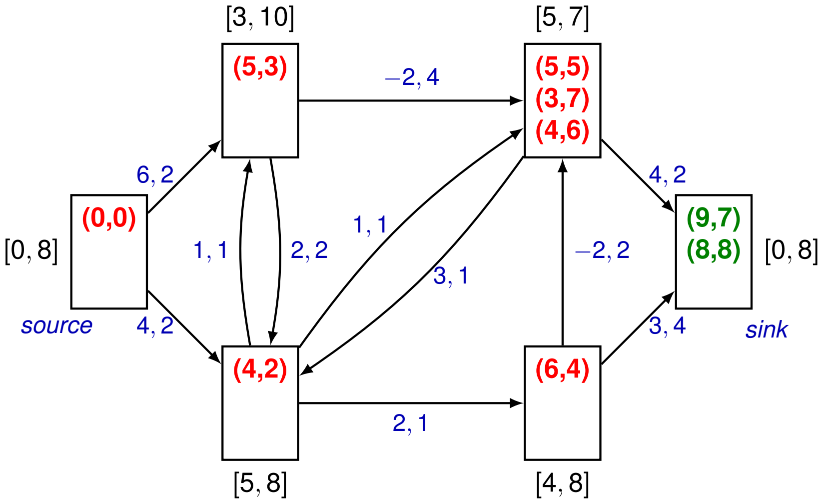
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



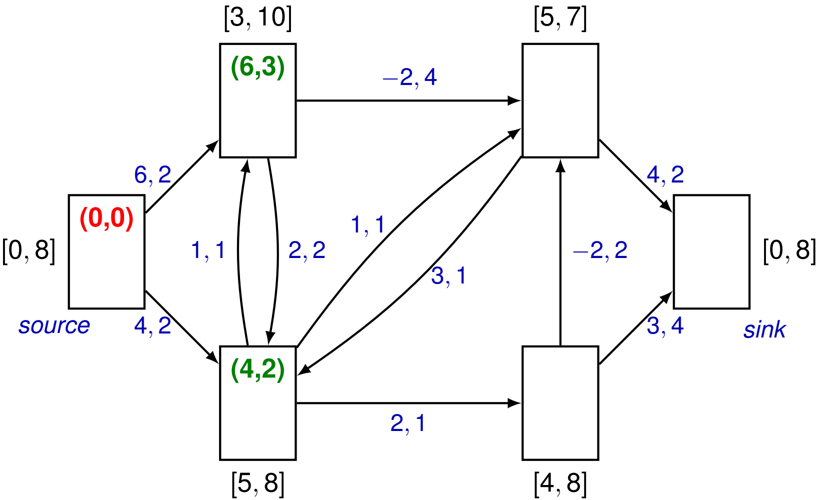
Basic labelling algorithm: label-setting example

Every label $L = (c^L, q_1^L)$



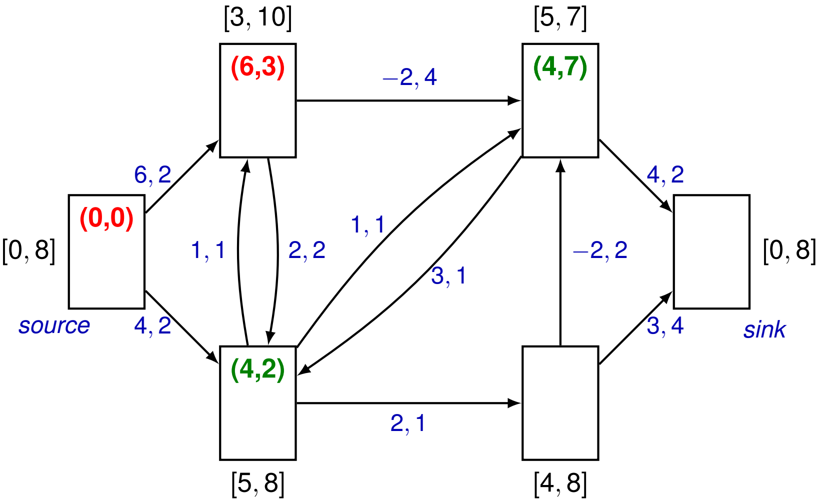
Basic labelling algorithm: label-correcting example

Every label $L = (c^L, q_1^L)$



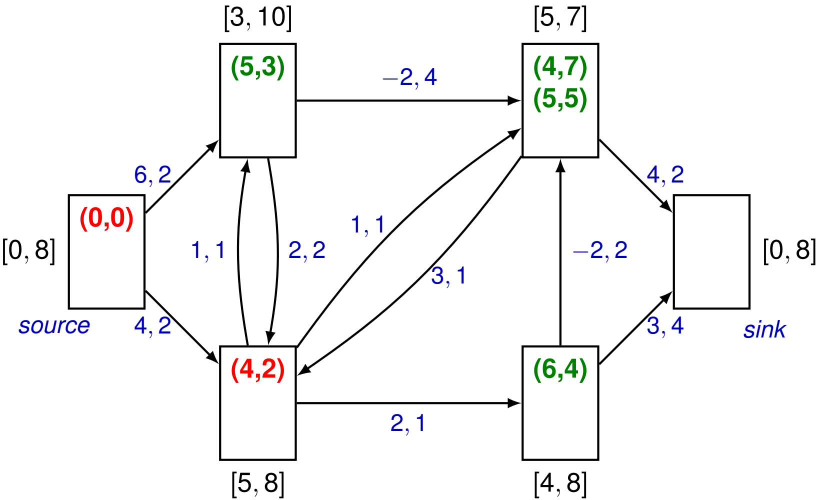
Basic labelling algorithm: label-correcting example

Every label $L = (c^L, q_1^L)$



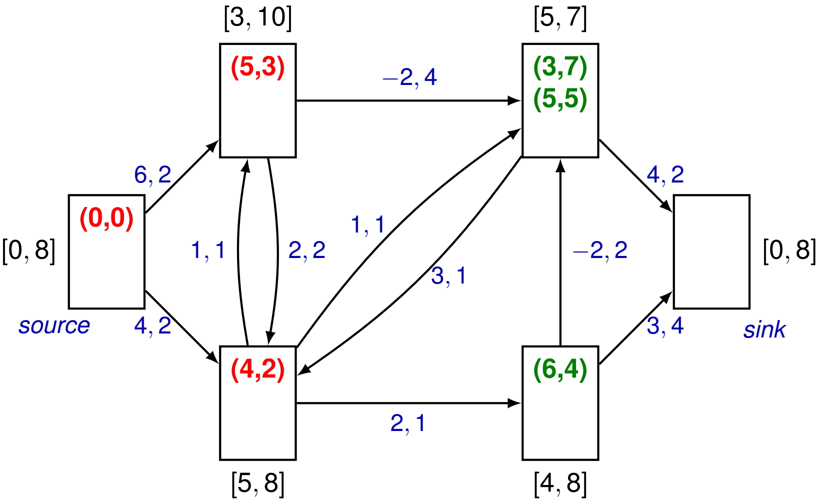
Basic labelling algorithm: label-correcting example

Every label $L = (c^L, q_1^L)$



Basic labelling algorithm: label-correcting example

Every label $L = (c^L, q_1^L)$



Literature: “embedded” algorithms for the RC(E)SPP

Almost all approaches are variants of the labelling algorithm

- Keep track of vertices which cannot be visited instead of visited vertices in a label [Feillet et al., 2004]
- Bi-directional search [Righini and Salani, 2006]
- Limited dominance checks by discretisation of the resource consumption [Fukasawa et al., 2006]

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004).
An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems.
Networks, 44(3):216–229.

Righini, G. and Salani, M. (2006).
Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints.
Discrete Optimization, 3(3):255 – 273.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F. (2006).
Robust branch-and-cut-and-price for the capacitated vehicle routing problem.
Mathematical Programming, 106(3):491–511.

Non-elementary relaxations of the pricing problem

Weakens the column generation lower bound,
but keeps the BCP correct

- *q*-routes [Christofides et al., 1981]
- *k*-cycle elimination [Irnich and Villeneuve, 2006]
  (too expensive for $k \geq 5$)
- *ng*-routes [Baldacci et al., 2011]

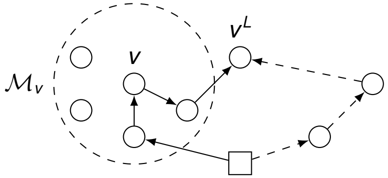
Non-elementary relaxations of the pricing problem

Weakens the column generation lower bound,
but keeps the BCP correct

- q-routes [Christofides et al., 1981]
- k-cycle elimination [Irnich and Villeneuve, 2006]
  (too expensive for $k \geq 5$)
- ng-routes [Baldacci et al., 2011]

For each vertex $v \in V$, define a memory $\mathcal{M}_v$ of vertices which "remember" v.

If $v^L \notin \mathcal{M}_v$, v is removed from $\mathcal{V}^L$.
Sets $\mathcal{V}^L$ are smaller $\Rightarrow$
stronger domination



Decremental state-space relaxation [Martinelli et al., 2014] for even tighter bounds

Dynamic *ng*-route relaxation [Roberti and Mingozzi, 2014]

Instance	Elementary bound		Dynamic *ng* bound	
	Gap	Time	Gap	Time
R202	0.72%	18	0.72%	58
R203	0.45%	72	0.45%	64
R204	0.88%	133	0.88%	76
R206	1.03%	45	1.04%	68
R207	0.42%	128	0.49%	79
R208	1.28%	267	1.34%	148
R209	1.57%	42	1.57%	33
R210	1.23%	34	1.23%	52
R211	1.61%	77	1.62%	54
RC204	0.49%	323	0.54%	131
RC207	1.62%	43	1.62%	38
RC208	1.21%	442	1.22%	66
Average	0.89%	151	0.91%	68

Table: Elementary bound [Lozano et al., 2016] vs. dynamic *ng* bound (hardest Solomon VRPTW instances)

Structure of RCSPP instances we want to solve

- A directed graph $G = (V, A)$.
- Unrestricted in sign reduced costs $\bar{c}_a$ on arcs $a \in A$
- Set R of "global" resources (usually one or two).
- Non-integer resource consumption $q_{a,r}$, $r \in R$, and accumulated resource consumption bounds $[l_{a,r}, u_{a,r}]$, $r \in R$, on arcs $a \in A$
- Up to $\approx$ 1000 of (more or less) local binary or (small) integer resources
- For simplicity, we suppose bijection between nodes and packing sets

We want to

Find a walk from the source to the sink minimizing the total reduced cost respecting the resource constrains, as well as many other (50–1000) different near-optimal feasible walks

Contents

Our approach to improve the labelling algorithm

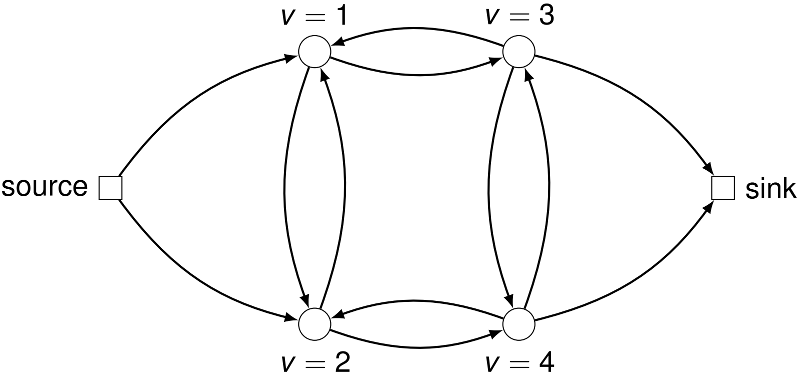
To our knowledge, no (published) attempts to

reduce the number of dominance checks

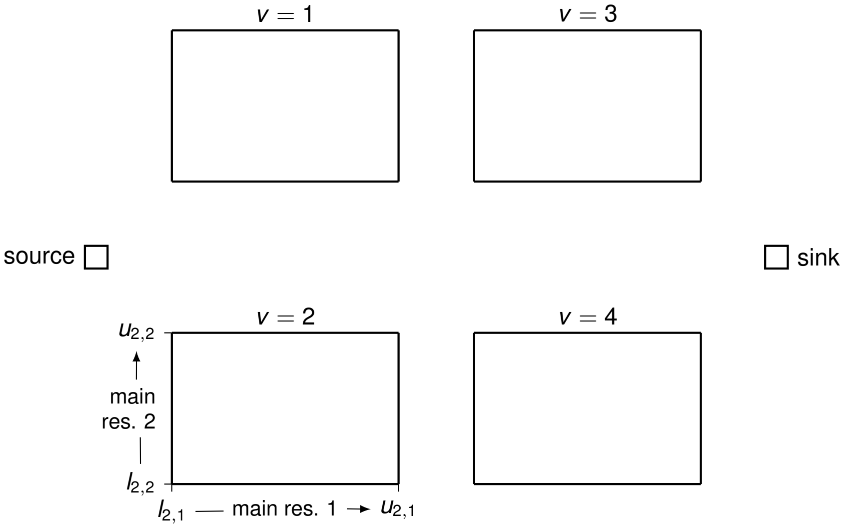
while keeping the dominance strength

in a labelling algorithm

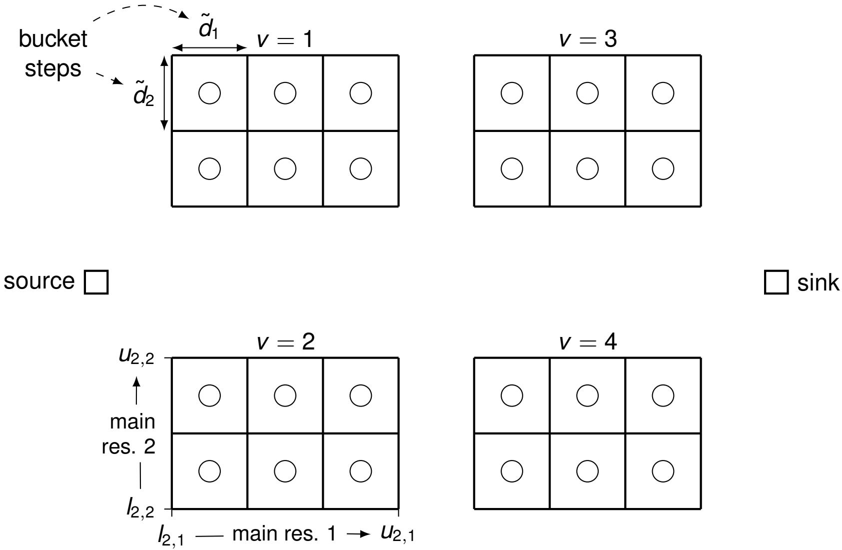
Original graph



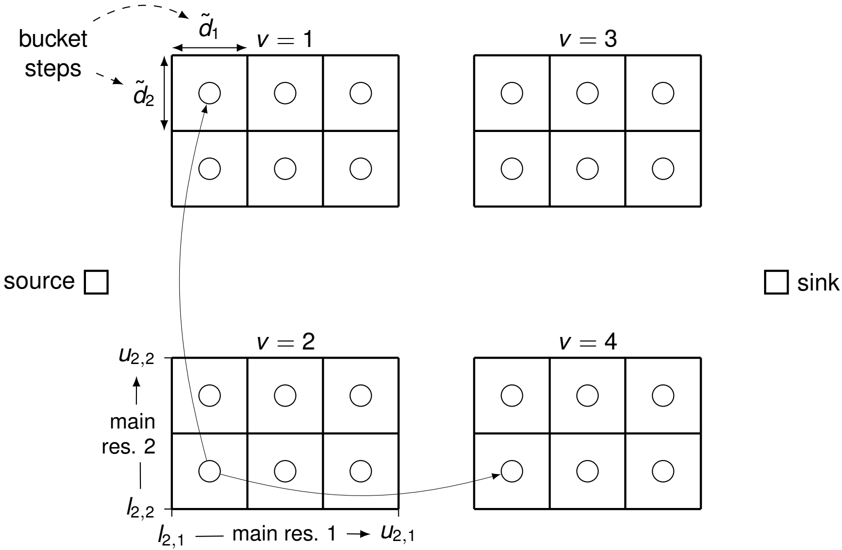
The bucket graph (with two main resources)



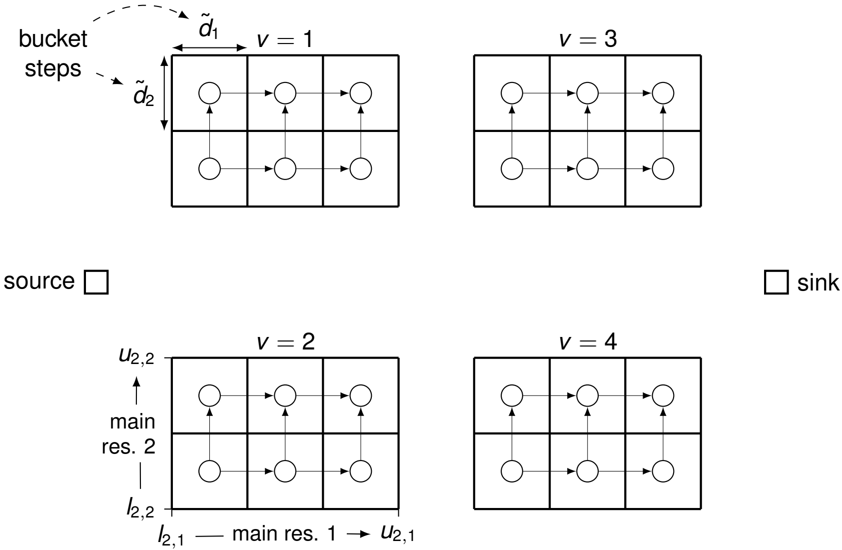
The bucket graph (with two main resources)



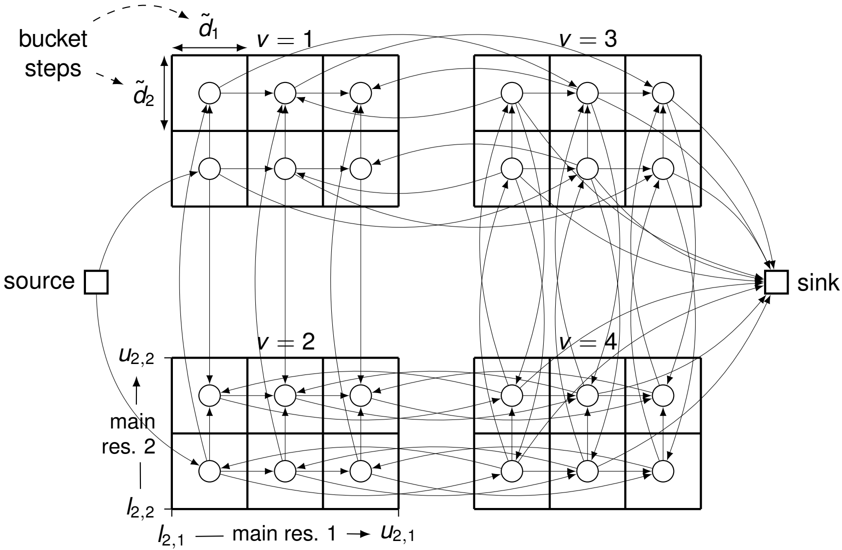
The bucket graph (with two main resources)



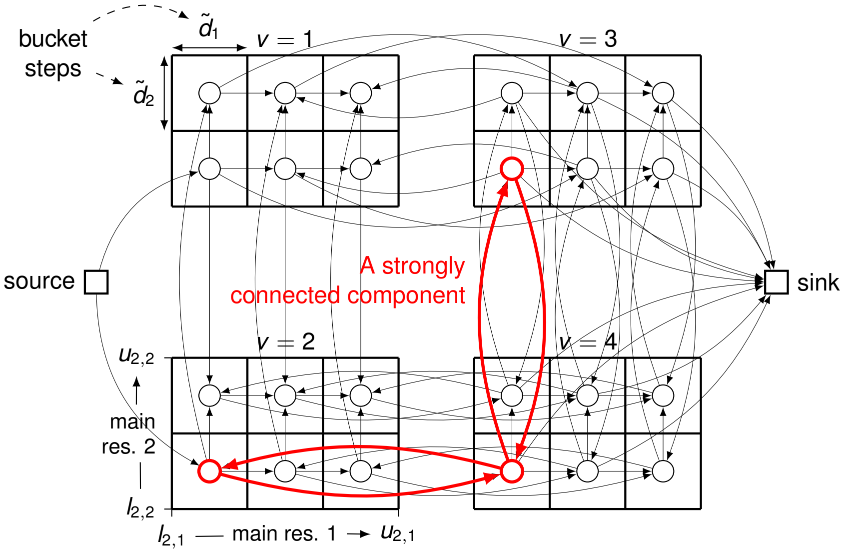
The bucket graph (with two main resources)



The bucket graph (with two main resources)



The bucket graph (with two main resources)



Extension order of labels

Extend labels according to a topological order of strongly connected components in the bucket graph.

Impact of bucket steps

Large enough bucket steps produce the standard label-correcting algorithm

- One bucket per vertex
- Bucket graph reduces to the original graph

Small enough bucket steps produce a label-setting algorithm

- Acyclic bucket graph
- Guarantee that only non-dominated labels are extended

Optimization of dominance checks

Practical observation

Higher dominance probability between labels with similar global resource consumption

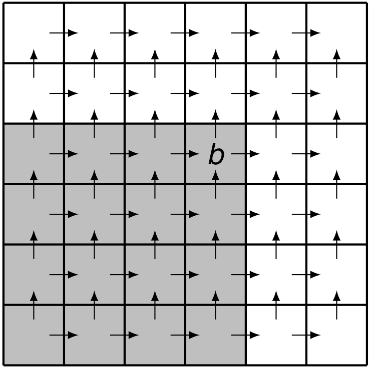
After the label's creation

check dominance with labels in the same bucket only!

Before the label's extension

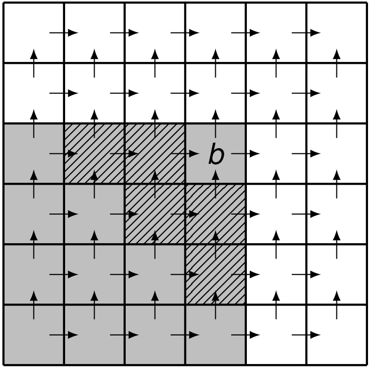
check dominance with labels in other buckets using bounds

Using bounds to reduce dominance checks between buckets



$\bar{c}_b^{best}$ — minimum reduced cost of labels in buckets $b' \preceq b$ (area )

Using bounds to reduce dominance checks between buckets



$\bar{c}_b^{best}$ — minimum reduced cost of labels in buckets $b' \preceq b$ (area )

Label L may be dominated in buckets $b' \preceq b$ only if $\bar{c}^L \geq \bar{c}_b^{best}$

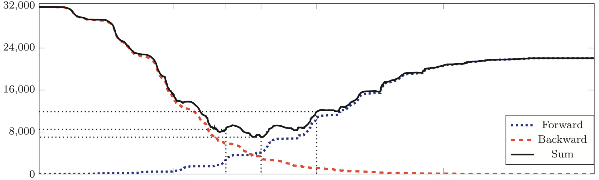
(only buckets in area  are tested)

Bi-directional variant of our algorithm

- Pick the first main resource and a threshold q_1^*
- In the forward labelling, keep only labels $\vec{L}$ with $q_1^{\vec{L}} \leq q_1^*$
- In the backward labelling, keep only labels $\overleftarrow{L}$ with $q_1^{\overleftarrow{L}} > q_1^*$
- Perform the concatenation step: a forward label $\vec{L}$ and a backward label $\overleftarrow{L}$ can be concatenated along arc $(v^{\vec{L}}, v^{\overleftarrow{L}})$
- Concatenation is accelerated using bounds $\bar{c}_{\overleftarrow{b}}^{best}$: if

$$\bar{c}^{\vec{L}} + \bar{c}_{(v^{\vec{L}}, v_{\overleftarrow{b}})} + \bar{c}_{\overleftarrow{b}}^{best} \geq UB(\bar{c}^*)$$

  then we can skip backward buckets $\overleftarrow{b}' \preceq \overleftarrow{b}$ while searching for a concatenation pair for label $\vec{L}$.
- Picture from [Tilk et al., 2017]:



Exploiting symmetry

If

- all resource consumption bounds are the same $[l_{a,r}, u_{a,r}] = [0, Q_r]$, $\forall a \in A$, $\forall r \in R$,
- for each arc $a = (i, j) \in A$ there exists arc $a' = (j, i) \in A$ with the same resource consumption $q_{a'} = q$ and the same reduced cost $\bar{c}_{a'} = \bar{c}_{a'}$,

then

- we can set $q_1^* = Q_1/2$
- and skip the backward labelling.

Contents

Computational impact of buckets steps

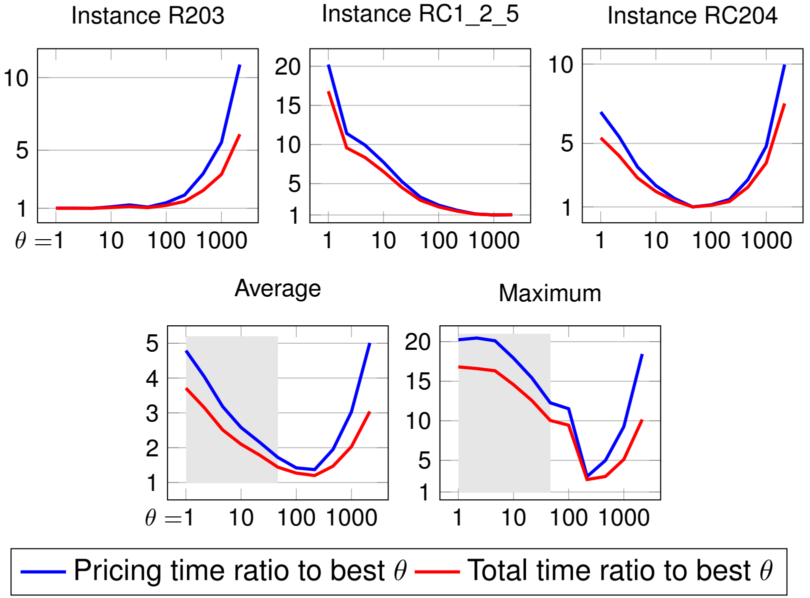
- 14 hardest [Solomon, 1987] instances with 100 customers and 60 [Gehring and Homberger, 2002] instances with 200 customers
- A full-blown state-of-the-art column-and-cut generation at the root (stop when the target lower bound is reached)
- We test the parameter θ — the maximum number of buckets per vertex:

$$\tilde{d}_1 = \frac{W}{\sqrt{\theta}}, \quad \tilde{d}_2 = \frac{u_{depot} - l_{depot}}{\sqrt{\theta}} \quad \text{(two global resources)}$$

$$\tilde{d} = \frac{u_{depot} - l_{depot}}{\theta} \quad \text{(one global resource)}$$

- $\theta = 1$ — standard label-correcting algorithm

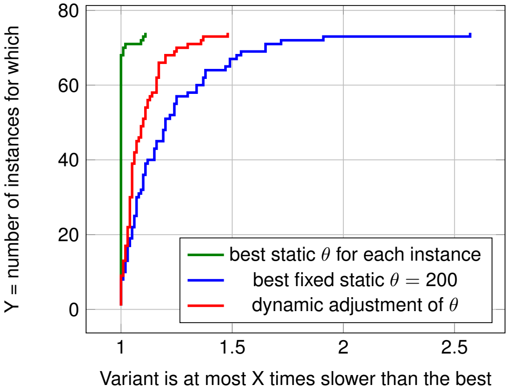
Computational impact of buckets steps



Dynamic adjustment of bucket steps

- Start with $\theta = 25$
- Multiply θ by 2 each time this ratio is above a threshold

$$\frac{\text{\# of dominance checks inside buckets}}{\text{\# of non-dominated labels}}$$



Contents

Arc elimination using path-reduced costs [Irnich et al., 2010]

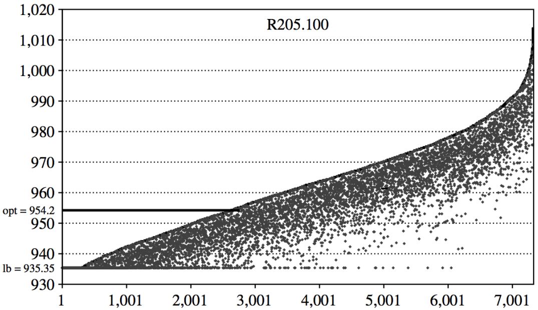
- Z_{RM} — optimum value of the master which gives the lower bound
- Z_{inc} — value of the incumbent integer solution
- $Z_{pricing}(a)$ — optimum solution value of the pricing problem solution, arc a being fixed to 1
- Arc a can be removed from the graph (it cannot take part of any improving solution) if

$$Z_{RM} + Z_{pricing}(a) \geq Z_{inc}$$

Arc elimination using path-reduced costs [Irnich et al., 2010]

- Z_{RM} — optimum value of the master which gives the lower bound
- Z_{inc} — value of the incumbent integer solution
- $Z_{pricing}(a)$ — optimum solution value of the pricing problem solution, arc *a* being fixed to 1
- Arc *a* can be removed from the graph (it cannot take part of any improving solution) if

$$Z_{RM} + Z_{pricing}(a) \geq Z_{inc}$$

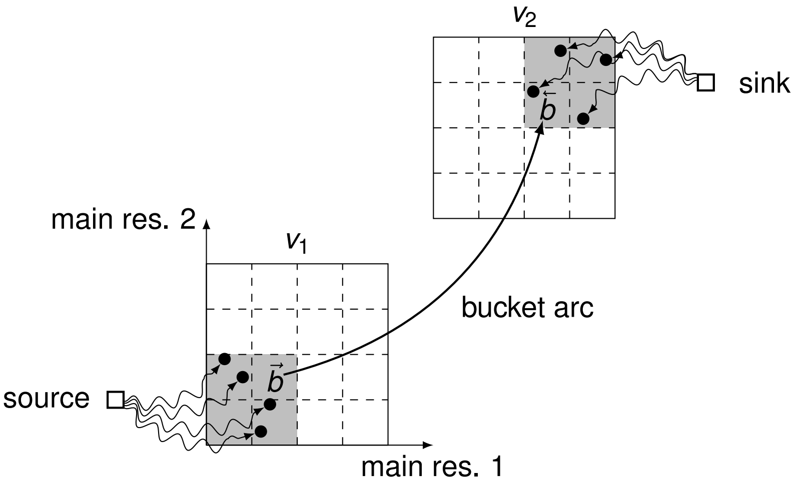


A good heuristic is very important!

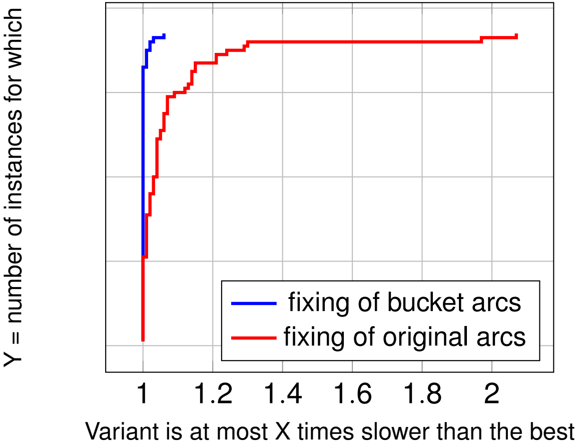
Bucket arc elimination using reduced costs

A sufficient condition to remove a bucket arc $(\vec{b}, (v_1, v_2), \overleftarrow{b})$

No pair of labels $(\vec{L}, \overleftarrow{L})$, $v^{\vec{L}} = v_1$, $v^{\overleftarrow{L}} = v_2$, $\vec{b}^{\vec{L}} \preceq \vec{b}$, $\overleftarrow{b}^{\overleftarrow{L}} \preceq \overleftarrow{b}$, producing a path by concatenation along arc (v_1, v_2) with reduced cost smaller than the current primal-dual gap.



Computational impact of bucket arc elimination (the root node only)



Bucket arc elimination: notes

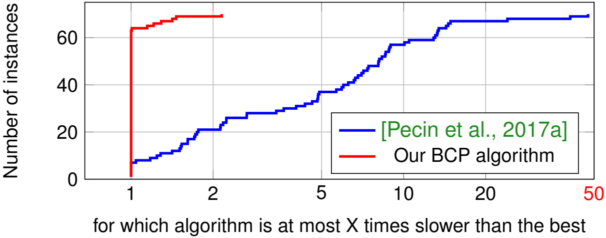
- Both forward and backward labelling should be performed completely, and not only until the “middle” point
- Arc elimination is much more expensive than the bi-directional labelling
- We use exhaustive completion bounds: $\vec{L}$ is extended only if there exists a label $\overleftarrow{L}$ such that its concatenation with the extension results in a path with the reduced cost smaller than the current primal-dual gap.
- values $\bar{c}^{best}_{\overleftarrow{b}}$ are used to speed-up the search for such label $\overleftarrow{L}$.

Contents

Computatonal results for classic VRPTW instances

14 hardest [Solomon, 1987] instances with 100 customers
60 [Gehring and Homberger, 2002] instances with 200 customers

Algorithm	Solved	65 instances solved by both	
		Aver. time (m)	Geom. time (m)
[Pecin et al., 2017a]	65/74	217.8	32.6
Our BCP algorithm	70/74	72.5	8.3



Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017).
New enhancements for the exact solution of the vehicle routing problem with time windows.
INFORMS Journal on Computing, 29(3):489–502.

Computatonal results for the MDVRP instances

Classic distance constrained multi-depot instances by [Cordeau et al., 1997] with up to 288 customers.

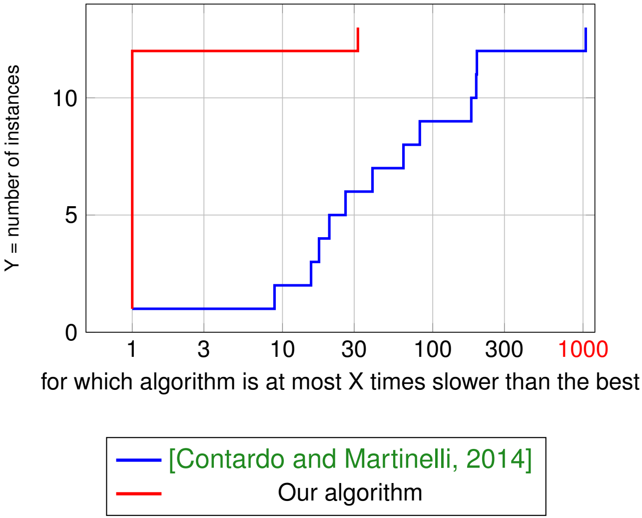
Algorithm	Solved	10 inst. solved by both	
		Aver. time	Geom. time
[Contardo and Martinelli, 2014]	10/13	269.8	8.4
Our algorithm	**22/22**	2.5	0.5

One improved BKS (instance “pr10”) over [Vidal et al., 2012]

Contardo, C. and Martinelli, R. (2014).
A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints.
Discrete Optimization, 12:129 – 146.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012).
A hybrid genetic algorithm for multidepot and periodic vehicle routing problems.
Operations Research, 60(3):611–624.

Computatonal results for the MDVRP instances: performance profile



Computatonal results for other problems

First exact algorithm for these vehicle routing variants

DCVRP Classic distance-constrained CVRP instances [Christofides et al., 1979]

SDVRP Standard distance-constrained site-dependent instances [Cordeau and Laporte, 2001]

HFVRP "Nightmare" heterogeneous fleet VRP instances (very large capacities) [Duhamel et al., 2011]

Class	Solved	Largest solved n	Smallest unsolved n	Geomean time	Improv. BKS
DCVRP	6/7	200	120	16m44s	0/7
SDVRP	7/10	216	240	11m26s	4/10
HFVRP	56/96	186	107	23m07s	43/96

Christofides, N., Mingozzi, A., and Toth, P. (1979).
Combinatorial Optimization, chapter "The vehicle routing problem", p. 315–338.
Wiley, Chichester.

Conclusions

- No universally best algorithm for the RCSPP, very different instances are considered in the literature
- Our approach is good for RCSPP instances coming from state-of-the-art Branch-Cut-and-Price algorithms for vehicle routing
- Bucket steps size is a critical instance-dependent parameter for the labelling algorithm
- Bucket arc elimination using reduced costs is possible and may be used by default (does not hurt)
- Significant computational improvement over the state-of-the-art for exact solution of important vehicle routing problems
- A generalization of our approach has been implemented in VRPSolver [Pessoa et al., 2019]

References I

Baldacci, R., Mingozzi, A., and Roberti, R. (2011).
New route relaxation and pricing strategies for the vehicle routing problem.
Operations Research, 59(5):1269–1283.

Christofides, N., Mingozzi, A., and Toth, P. (1979).
Combinatorial Optimization, chapter The vehicle routing problem, pages 315–338.
Wiley, Chichester.

Christofides, N., Mingozzi, A., and Toth, P. (1981).
Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations.
Mathematical Programming, 20(1):255–282.

Contardo, C. and Martinelli, R. (2014).
A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints.
Discrete Optimization, 12:129 – 146.

References II

Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997).
A tabu search heuristic for periodic and multi-depot vehicle routing problems.
Networks, 30(2):105–119.

Cordeau, J.-F. and Laporte, G. (2001).
A tabu search algorithm for the site dependent vehicle routing problem with time windows.
INFOR: Information Systems and Operational Research, 39(3):292–298.

Duhamel, C., Lacomme, P., and Prodhon, C. (2011).
Efficient frameworks for greedy split and new depth first search split procedures for routing problems.
Computers and Operations Research, 38(4):723 – 739.

Dumitrescu, I. and Boland, N. (2003).
Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem.
Networks, 42(3):135–153.

References III

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004).
An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems.
Networks, 44(3):216–229.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F. (2006).
Robust branch-and-cut-and-price for the capacitated vehicle routing problem.
Mathematical Programming, 106(3):491–511.

Gehring, H. and Homberger, J. (2002).
Parallelization of a two-phase metaheuristic for routing problems with time windows.
Journal of Heuristics, 8(3):251–276.

Irnich, S., Desaulniers, G., Desrosiers, J., and Hadjar, A. (2010).
Path-reduced costs for eliminating arcs in routing and scheduling.
INFORMS Journal on Computing, 22(2):297–313.

References IV

Irnich, S. and Villeneuve, D. (2006).
The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$.
INFORMS Journal on Computing, 18(3):391–406.

Lozano, L., Duque, D., and Medaglia, A. L. M. L. (2016).
An exact algorithm for the elementary shortest path problem with resource constraints.
Transportation Science, 50(1):348–357.

Lozano, L. and Medaglia, A. L. (2013).
On an exact method for the constrained shortest path problem.
Computers & Operations Research, 40(1):378 – 384.

Martinelli, R., Pecin, D., and Poggi, M. (2014).
Efficient elementary and restricted non-elementary route pricing.
European Journal of Operational Research, 239(1):102 – 111.

References V

Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017a).
New enhancements for the exact solution of the vehicle routing problem with time windows.
INFORMS Journal on Computing, 29(3):489–502.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017b).
Improved branch-cut-and-price for capacitated vehicle routing.
Mathematical Programming Computation, 9(1):61–100.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2017).
Automation and combination of linear-programming based stabilization techniques in column generation.
INFORMS Journal on Computing, accepted.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2019).
A generic exact solver for vehicle routing and related problems.
In Lodi, A. and Nagarajan, V., editors, *Integer Programming and Combinatorial Optimization*, volume 11480 of *Lecture Notes in Computer Science*, pages 354–369, Cham. Springer International Publishing.

References VI

Righini, G. and Salani, M. (2006).
Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints.
Discrete Optimization, 3(3):255 – 273.

Roberti, R. and Mingozzi, A. (2014).
Dynamic ng-path relaxation for the delivery man problem.
Transportation Science, 48(3):413–424.

Santos, L., ao Coutinho-Rodrigues, J., and Current, J. R. (2007).
An improved solution algorithm for the constrained shortest path problem.
Transportation Research Part B: Methodological, 41(7):756 – 771.

Sedeno-Noda, A. and Alonso-Rodríguez, S. (2015).
An enhanced k-sp algorithm with pruning strategies to solve the constrained shortest path problem.
Applied Mathematics and Computation, 265:602 – 618.

References VII

Solomon, M. M. (1987).
Algorithms for the vehicle routing and scheduling problems with time window constraints.
Operations Research, 35(2):254–265.

Thomas, B. W., Calogiuri, T., and Hewitt, M. (2019).
An exact bidirectional a* approach for solving resource-constrained shortest path problems.
Networks, 73(2):187–205.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017).
Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster.
European Journal of Operational Research, 261(2):530 – 539.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012).
A hybrid genetic algorithm for multidepot and periodic vehicle routing problems.
Operations Research, 60(3):611–624.

References VIII

Zhu, X. and Wilhelm, W. E. (2012).
A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation.
Computers & Operations Research, 39(2):164 – 178.