A Bucket Graph Based Labelling Algorithm for the Resource Constrained Shortest Path Problem with Applications to Vehicle Routing

Ruslan Sadykov¹,² Artur Pessoa³ Eduardo Uchoa³

¹ Inria Bordeaux, France
² Université Bordeaux, France
³ Universidade Federal Fluminense, Brazil

IFORS 2017
Quebec city, Canada, July 20
Computational results for classic VRPTW instances

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Solved</th>
<th>65 instances solved by both</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Aver. time (m)</td>
</tr>
<tr>
<td>[Pecin et al., 2017a]</td>
<td>65/74</td>
<td>217.8</td>
</tr>
<tr>
<td>Our BCP algorithm</td>
<td>70/74</td>
<td>72.5</td>
</tr>
</tbody>
</table>

Number of instances

for which algorithm is at most X times slower than the best

---

Branch-Cut-and-Price algorithm components contributing to improvement over [Pecin et al., 2017a]

- New bucket graph based labelling algorithm for the Resource Constrained Shortest Path pricing problem
- Automatic dual price smoothing stabilization [Pessoa et al., 2017]
- Dynamic ng-path relaxation [Roberti and Mingozzi, 2014]

---


Structure of RCSPP instances we want to solve

- A complete directed graph $G = (V, A)$.
- Unrestricted in sign reduced costs $\bar{c}_a$ on arcs $a \in A$.
- Two global (capacity and time) resources with non-negative non-integer resource consumption $d_{a,1}, d_{a,2}$ on arcs $a \in A$.
- Resource consumption bounds $[0, W]$ and $[l_v, u_v]$ on vertices $v \in V$.
- Up to $\approx 500 – 1000$ of (more or less) local binary or (small) integer resources.

We want to

Find a walk from the sources to the sink minimizing the total reduced cost respecting the resource constrains, as well as many other (up to 1000) different near-optimal feasible walks.
Literature: “standalone” algorithms for the RCSPP

Test instances with an acyclic sparse graph with global, but few 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] or the $k$-shortest paths problem
  [Santos et al., 2007]
- Pulse Algorithm (limited dominance and depth-first search)
  [Lozano and Medaglia, 2013]
  - the best “standalone” algorithm
  - fails completely for our hard instances [Pecin, 2014]
Basic labelling algorithm

\[ L = \bigcup_{v \in V} L_v \] — set of non-extended labels

\[ E = \bigcup_{v \in V} E_v \] — set of extended labels

\[ L \rightarrow \{(\text{source}, 0, 0, 0, \{\text{source}\})\}, \ E \leftarrow \emptyset \]

while \( L \neq \emptyset \) do

pick a label \( L \) in \( L \), \( v^L \neq \text{sink} \)

\[ L \leftarrow L \setminus \{L\}, \ E \leftarrow E \cup \{L\} \]

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 \( L_v \cup E_v \) then

\[ L \leftarrow L \cup \{L'\} \]

remove from \( L_v \cup E_v \) all labels dominated by \( L' \)

return a label in \( L_{\text{sink}} \) 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 \( L_v \))
Literature: “embedded” algorithms for the RCSPP

All approaches are variants of the labelling algorithm

- Bi-directional search [Righini and Salani, 2006]
- Enumeration of elementary routes using completion bounds from the *ng-path relaxation* [Baldacci et al., 2011]
- Completion bounds from *dynamic state-space relaxation* of the resources from non-robust cuts [Contardo and Martinelli, 2014]
- Limited dominance checks by *discretisation* of the resource consumption [Pecin et al., 2017b].

---


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)

\[
v = 1
\]

\[
v = 2
\]

\[
v = 3
\]

\[
v = 4
\]

source \(\square\)

sink \(\square\)
The bucket graph (with two main resources)
The bucket graph (with two main resources)

bucket steps

source

sink

bucket

time window

capacity

0 ————> W
The bucket graph (with two main resources)

bucket steps

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]
The bucket graph (with two main resources)

bucket steps

\[ v = 1 \]
\[ v = 2 \]
\[ v = 3 \]
\[ v = 4 \]

source

sink

time window

\[ l_2 \]

\[ u_2 \]

\[ \tilde{d}_1 \]

\[ \tilde{d}_2 \]

\[ 0 \rightarrow \text{capacity} \rightarrow W \]
The bucket graph (with two main resources)

A strongly connected component
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
- One strongly connected component (for our instances)

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

\[ \bar{c}_b^{\text{best}} \] — minimum reduced cost of labels in buckets \( b' \leq b \) (area)

Label \( L \) may be dominated in buckets \( b' \leq b \) only if \( \bar{c}^L \geq \bar{c}_b^{\text{best}} \)

(only buckets in area are tested)
Bi-directional variant

- Pick a global resource (f.e. capacity) and a threshold $w^*$
- In the forward labelling, keep only labels $\vec{L}$ with $w^{\vec{L}} \leq w^*$
- In the backward labelling, keep only labels $\vec{L}$ with $w^{\vec{L}} > w^*$
- Perform the concatenation step: a forward label $\vec{L}$ and a backward label $\vec{L}$ can be concatenated along arc $(v^{\vec{L}}, v^{\vec{L}^\dagger})$
- Concatenation is accelerated using bounds $\bar{c}_b^{\text{best}}$: if

$$\bar{c}^{\vec{L}} + \bar{c}_{(v^{\vec{L}}, v_b)} + \bar{c}_b^{\text{best}} \geq UB(\bar{c}^*)$$

then we can skip backward buckets $\vec{b'} \leq \vec{b}$ while searching for a concatenation pair for label $\vec{L}$

- Exploiting symmetry: if all time windows are the same, the backward labelling is equivalent to the forward, we use only forward buckets and labels in concatenation
Computational impact of buckets steps

- Same VRPTW instances
- 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 $\theta$ — the maximum number of buckets per vertex:

$$\tilde{d}_1 = \frac{W}{\sqrt{\theta}}, \quad \tilde{d}_2 = \frac{u_{depot} - l_{depot}}{\sqrt{\theta}}$$

(two global resources)

$$\tilde{d} = \frac{u_{depot} - l_{depot}}{\theta}$$

(one global resource)

$\theta = 1$ — standard label-correcting algorithm
Computational impact of buckets steps

Instance R203

\[ \theta = 1 \quad 10 \quad 100 \quad 1000 \]

Instance RC1_2_5

\[ \theta = 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

Instance RC204

\[ \theta = 1 \quad 10 \quad 100 \quad 1000 \]

Average

Maximum

\[ \theta = 1 \quad 10 \quad 100 \quad 1000 \]

\[ \theta = 1 \quad 10 \quad 100 \quad 1000 \]

Pricing time ratio to best \( \theta \)

Total time ratio to best \( \theta \)
Dynamic adjustment of bucket steps

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

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

Variant is at most $X$ times slower than the best

- $Y = \text{number of instances for which}$
- Best static $\theta$ for each instance
- Best fixed static $\theta = 200$
- Dynamic adjustment of $\theta$

![Graph showing performance comparison between methods]
Fixing of bucket arcs by reduced cost

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

No pair of labels \((\vec{L}, \vec{L}), v^L = v_1, v^L = v_2, \vec{b}^L \leq \vec{b}, \vec{b}^L \leq \vec{b}\)

producing a path by concatenation along arc \((v_1, v_2)\) with reduced cost smaller than the current primal-dual gap.
Computational impact of fixing bucket arcs by reduced cost (the root node only)

Variant is at most $X$ times slower than the best

Y = number of instances for which

- fixing of bucket arcs
- fixing of original arcs

Variant is at most $X$ times slower than the best
Computational results for the MDVRP instances

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Solved</th>
<th>10 inst. solved by both</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Aver. time</td>
</tr>
<tr>
<td>[Contardo and Martinelli, 2014]</td>
<td>10/13</td>
<td>269.8</td>
</tr>
<tr>
<td>Our algorithm</td>
<td>22/22</td>
<td>2.5</td>
</tr>
</tbody>
</table>

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

A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints.  

A hybrid genetic algorithm for multidepot and periodic vehicle routing problems.  
Computational results for the MDVRP instances: performance profile

For which algorithm is at most $X$ times slower than the best

$Y = \text{number of instances}$

[Contardo and Martinelli, 2014]

Our algorithm
Computational 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]

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

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
- Fixing bucket arcs by 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
References I


On an exact method for the constrained shortest path problem.  

Improved exact algorithms for the vehicle routing problem.  
GERAD seminar presentation.

New enhancements for the exact solution of the vehicle routing problem with time windows.  

Improved branch-cut-and-price for capacitated vehicle routing.  
References IV


