Mathematical Optimization and Polyhedral Approaches: Column Generation, Branch and Price and Cut

Master EUR Maths & Computer Science
Université Sorbonne Paris Nord

Lucas Létocart

lucas.letocart@lipn.univ-paris13.fr

LIPN (Laboratoire d'Informatique de Paris Nord)
Université Sorbonne Paris Nord & CNRS
src: Stefan Irnich

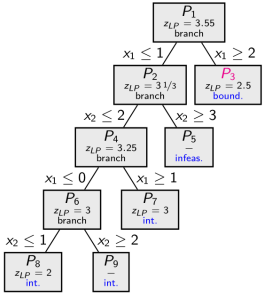
Agenda

What is Column Generation and a Branch-and-Price-and-Cut Algorithm?

- Two examples — Tutorial
- Dantzig-Wolfe decomposition
- Branch-and-Price-and-Cut

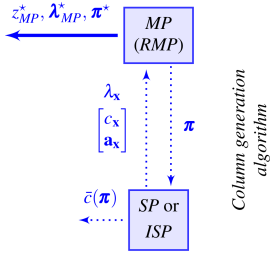
What is a Branch-Price-and-Cut Algorithm?

- It is a Branch-and-Bound algorithm in which



What is a Branch-Price-and-Cut Algorithm?

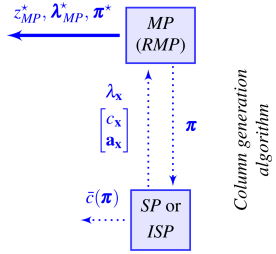
- It is a Branch-and-Bound algorithm in which
- an LP—the master program (MP)—is solved with Column Generation in each node of the branch-and-bound tree
  - MP has a very large number of variables (=columns)



from: (Desrosiers *et al.*, 2024, p. 49)

What is a Branch-Price-and-Cut Algorithm?

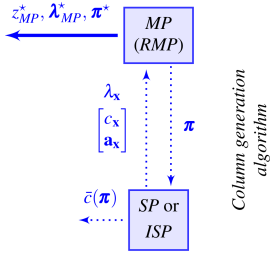
- It is a Branch-and-Bound algorithm in which
- an LP—the master program (MP)—is solved with Column Generation in each node of the branch-and-bound tree
  - MP has a very large number of variables (=columns)
  - Iterative procedure that alternates between solving a restricted master problem (RMP) and a pricing subproblem (SP)



from: (Desrosiers *et al.*, 2024, p. 49)

What is a Branch-Price-and-Cut Algorithm?

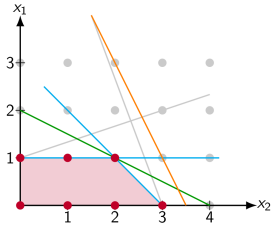
- It is a Branch-and-Bound algorithm in which
- an LP—the master program (MP)—is solved with Column Generation in each node of the branch-and-bound tree
  - MP has a very large number of variables (=columns)
  - Iterative procedure that alternates between solving a restricted master problem (RMP) and a pricing subproblem (SP)
  - Stops when no more negative reduced-cost variables are found



from: (Desrosiers *et al.*, 2024, p. 49)

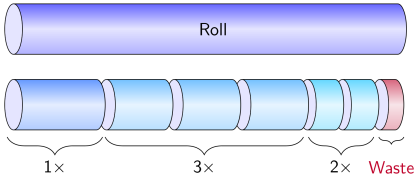
What is a Branch-Price-and-Cut Algorithm?

- It is a Branch-and-Bound algorithm in which
- an LP—the master program (MP)—is solved with Column Generation in each node of the branch-and-bound tree
  - MP has a very large number of variables (=columns)
  - Iterative procedure that alternates between solving a restricted master problem (RMP) and a pricing subproblem (SP)
  - Stops when no more negative reduced-cost variables are found
- and Cutting Planes are added to strengthen the linear relaxation (at some nodes).

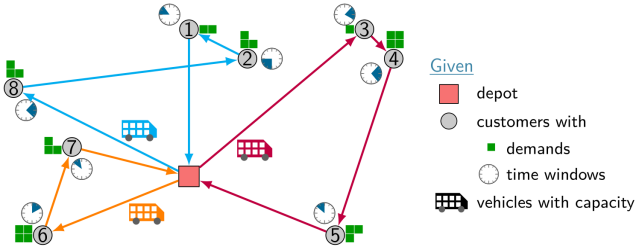


Two Examples

- Cutting and Packing
  →One-dimensional cutting stock problem (CS) and bin packing problem (BP)



- Vehicle Routing
  →The vehicle routing problem with time windows (VRPTW)

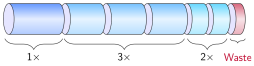


Gilmore and Gomory Formulation

Given:

- Bin size/width C
- (Types of) items $I = \{1, 2, \ldots, m\}$ with weight $w_i \leq C$ (and demand $d_i \in \mathbb{N}$) for $i \in I$

Find: Packing of items into a minimum number of bins
(Plan production of demanded items minimizing row material)

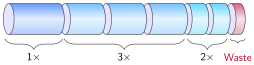


Gilmore and Gomory Formulation

Given:

- Bin size/width C
- (Types of) items $I = \{1, 2, \ldots, m\}$ with weight $w_i \leq C$ (and demand $d_i \in \mathbb{N}$) for $i \in I$

Find: Packing of items into a minimum number of bins
(Plan production of demanded items minimizing row material)



Gilmore and Gomory (1961) formulation:

- Set P of feasible patterns $\mathrm{p} = (p_i)_{i \in I}$
- For a pattern $\mathrm{p} \in P$, it must be possible to pack $p_i \in \mathbb{Z}_+$ items of type $i \in I$ together into a single bin i.e., $\sum_i w_i p_i \leq C$

For each $\mathrm{p} \in P$, the variable λ_p indicates how often the pattern p is used:

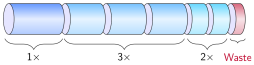
$$
\begin{aligned}
z(P) = \min & \sum_{\mathrm{p} \in P} 1 \cdot \lambda_\mathrm{p} & & \\
\text{subject to} \quad & \sum_{\mathrm{p} \in P} p_i \lambda_\mathrm{p} \geq d_i, & & i \in I, \\
& \lambda_\mathrm{p} \in \mathbb{Z}_+, & & \mathrm{p} \in P.
\end{aligned}
$$

Gilmore and Gomory Formulation

Given:

- Bin size/width C
- (Types of) items $I = \{1, 2, \ldots, m\}$ with weight $w_i \leq C$ (and demand $d_i \in \mathbb{N}$) for $i \in I$

Find: Packing of items into a minimum number of bins
(Plan production of demanded items minimizing row material)



Gilmore and Gomory (1961) formulation:

- Set P of feasible patterns $\mathrm{p} = (p_i)_{i \in I}$
- For a pattern $\mathrm{p} \in P$, it must be possible to pack $p_i \in \mathbb{Z}_+$ items of type $i \in I$ together into a single bin i.e., $\sum_i w_i p_i \leq C$

For each $\mathrm{p} \in P$, the variable λ_p indicates how often the pattern p is used:

$$
\begin{aligned}
z(P) = \min & \sum_{\mathrm{p} \in P} 1 \cdot \lambda_\mathrm{p} \\
\text{subject to} \quad & \sum_{\mathrm{p} \in P} p_i \lambda_\mathrm{p} \geq d_i, && i \in I, \\
& \lambda_\mathrm{p} \in \mathbb{Z}_+, && \mathrm{p} \in P.
\end{aligned}
$$

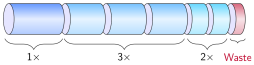
Linear relaxation (MP) replaces $\lambda_\mathrm{p} \in \mathbb{Z}_+$ by $\lambda_\mathrm{p} \geq 0$.

Gilmore and Gomory Formulation

Given:

- Bin size/width C
- (Types of) items $I = \{1, 2, \ldots, m\}$ with weight $w_i \leq C$ (and demand $d_i \in \mathbb{N}$) for $i \in I$

Find: Packing of items into a minimum number of bins
(Plan production of demanded items minimizing row material)



Gilmore and Gomory (1961) formulation:

- Set P of feasible patterns $\mathrm{p} = (p_i)_{i \in I}$
- For a pattern $\mathrm{p} \in P$, it must be possible to pack $p_i \in \mathbb{Z}_+$ items of type $i \in I$ together into a single bin i.e., $\sum_i w_i p_i \leq C$

For each $\mathrm{p} \in P$, the variable λ_p indicates how often the pattern p is used:

$$z(P) = \min \sum_{\mathrm{p} \in P} 1 \cdot \lambda_\mathrm{p} \qquad \text{duals:}$$

$$\text{subject to} \quad \sum_{\mathrm{p} \in P} p_i \lambda_\mathrm{p} \geq d_i, \qquad i \in I, \qquad (\pi_i)_{i \in I}$$

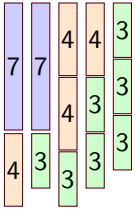
$$\lambda_\mathrm{p} \in \mathbb{Z}_+, \qquad \mathrm{p} \in P.$$

Linear relaxation (MP) replaces $\lambda_\mathrm{p} \in \mathbb{Z}_+$ by $\lambda_\mathrm{p} \geq 0$.

Example: Instance of **CS** with width $C = 11$, $I = \{1, 2, 3\}$ and items with $w_1 = 7, w_2 = 4, w_3 = 3$, and demand $d_1 = 125$, $d_2 = 310$, $d_3 = 100$.

Example: Instance of **CS** with width $C = 11$, $I = \{1, 2, 3\}$ and items with $w_1 = 7, w_2 = 4, w_3 = 3$, and demand $d_1 = 125$, $d_2 = 310$, $d_3 = 100$.

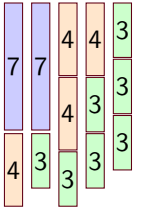
$$
\begin{aligned}
z(P_{CS}) \quad &= \quad \min \mathbb{1}^\top \lambda \\
\text{subject to} \quad & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} \lambda \geq \begin{bmatrix} 125 \\ 310 \\ 100 \end{bmatrix} \\
& \lambda \in \mathbb{Z}_+^5
\end{aligned}
$$



only maximal patterns

Example: Instance of **CS** with width $C = 11$, $I = \{1, 2, 3\}$ and items with $w_1 = 7, w_2 = 4, w_3 = 3$, and demand $d_1 = 125$, $d_2 = 310$, $d_3 = 100$.

$$
\begin{array}{rcl}
z(P_{CS}) & = & \min \mathbb{1}^\top \lambda \\
\text{subject to} & & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} \lambda \geq \begin{bmatrix} 125 \\ 310 \\ 100 \end{bmatrix} \\
 & & \lambda \in \mathbb{Z}^5_+
\end{array}
$$

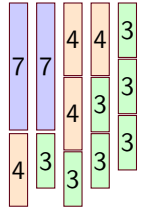


only maximal patterns

Example: Instance of **BP** with size $C = 10$, items $I = \{1, \ldots, 4\}$ of weight $w_1 = 5, w_2 = w_3 = w_4 = 2$.

Example: Instance of **CS** with width $C = 11$, $I = \{1, 2, 3\}$ and items with $w_1 = 7, w_2 = 4, w_3 = 3$, and demand $d_1 = 125$, $d_2 = 310$, $d_3 = 100$.

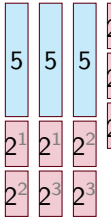
$$
\begin{aligned}
z(P_{CS}) \quad &= \quad \min \mathbb{1}^\top \lambda \\
\text{subject to} \quad & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} \lambda \geq \begin{bmatrix} 125 \\ 310 \\ 100 \end{bmatrix} \\
& \lambda \in \mathbb{Z}_+^5
\end{aligned}
$$



only maximal patterns

Example: Instance of **BP** with size $C = 10$, items $I = \{1, \ldots, 4\}$ of weight $w_1 = 5, w_2 = w_3 = w_4 = 2$.

$$
\begin{aligned}
z(P_{BP}) \quad &= \quad \min \mathbb{1}^\top \lambda \\
\text{subject to} \quad & \begin{bmatrix} 1 & 1 & 1 & 0 & \cdots \\ 1 & 1 & 0 & 1 & \cdots \\ 1 & 0 & 1 & 1 & \cdots \\ 0 & 1 & 1 & 1 & \cdots \end{bmatrix} \lambda = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& \lambda \in \mathbb{Z}_+^{|P|}
\end{aligned}
$$



. . . more but non-maximal patterns

Number of pattern grows quickly with m and C

E.g., $m = 5$, $C = 400$, $w = (10, 8, 5, 4, 3)$ $\Rightarrow$ 764631 maximal patters

Number of pattern grows quickly with m and C

E.g., $m = 5$, $C = 400$, $w = (10, 8, 5, 4, 3)$ $\Rightarrow$ 764631 maximal patters

The linear relaxation of the Gilmore and Gomory formulation can be solved by column generation.

Number of pattern grows quickly with m and C

E.g., $m = 5$, $C = 400$, $w = (10, 8, 5, 4, 3)$ $\Rightarrow$ 764631 maximal patters

The linear relaxation of the Gilmore and Gomory formulation can be solved by column generation.

The restricted master program (RMP) is a linear relaxation that contains a (small) subset $P' \subset P$ of all patterns of the Gilmore and Gomory formulation. It provides a generally fractional primal solution $(\bar{\lambda}_p)$ as well as dual prices $\pi = (\pi_i)_{i \in I}$.

Number of pattern grows quickly with m and C

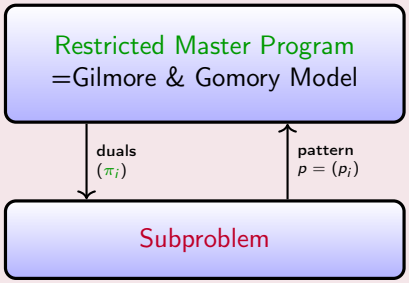
E.g., $m = 5$, $C = 400$, $w = (10, 8, 5, 4, 3)$ $\Rightarrow$ 764631 maximal patters

The linear relaxation of the Gilmore and Gomory formulation can be solved by column generation.

The restricted master program (RMP) is a linear relaxation that contains a (small) subset $P' \subset P$ of all patterns of the Gilmore and Gomory formulation. It provides a generally fractional primal solution $(\bar{\lambda}_p)$ as well as dual prices $\pi = (\pi_i)_{i \in I}$.

The column generation subproblem (a.k.a. pricing problem) is the problem of finding a feasible pattern $p = (p_i)_{i \in I} \in P$ with negative reduced cost:

$$\tilde{c}^\star(\pi) = \min_{p \in P} \left(1 - \sum_{i \in I} p_i \pi_i\right) = 1 - \left(\max_{p \in P} \sum_{i \in I} p_i \pi_i\right)$$



For CS and BP:

- Subproblem is variant of Knapsack Problem
- Solved by Dynamic Programming

For cutting stock **CS**:

$$z_{KP}(\pi) = \max \sum_{i \in I} \pi_i x_i$$

$$\text{subj. to} \sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \mathbb{Z}_+ \qquad \forall i \in I$$

For bin packing **BP**:

$$z_{KP}(\pi) = \max \sum_{i \in I} \pi_i x_i$$

$$\text{subj. to} \sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \{0, 1\} \qquad \forall i \in I$$

For cutting stock **CS**:

$$z_{KP}(\pi) = \max \sum_{i \in I} \pi_i x_i$$
$$\text{subj. to} \sum_{i \in I} w_i x_i \leq C$$
$$x_i \in \mathbb{Z}_+ \qquad \forall i \in I$$

For bin packing **BP**:

$$z_{KP}(\pi) = \max \sum_{i \in I} \pi_i x_i$$
$$\text{subj. to} \sum_{i \in I} w_i x_i \leq C$$
$$x_i \in \{0,1\} \qquad \forall i \in I$$

The new pattern $p = (p_i)_{i \in I}$ is then given by the solution values $\bar{x}_i$ of the KP, i.e.,

$$p_i := \bar{x}_i, \qquad i \in I.$$

It has negative reduced cost if and only if $z_{KP}(\pi) > 1$.

Initialization of the RMP

Example: Roll width $C = 102$, $m = 5$ pieces of width $w = (10, 8, 5, 4, 3)^\top$ with demand $b = (37, 920, 177, 422, 899)^\top$. approx. 5000 maximal patterns

Initial feasible solution: one pattern for each piece $i \in \{1, 2, \ldots, m\}$

1. Take piece i with maximal multiplicity $\lfloor C/w_i \rfloor$
2. Fill remaining "space" with smallest piece so that loss $< \min_k w_k$

Initial patterns:

$$\begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 12 \\ 0 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 20 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 25 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 34 \end{pmatrix}$$

Restricted master program (RMP):

$$
\begin{array}{lccccccll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq & 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq & 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq & 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & \geq & 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & \geq & 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941			

Restricted master program (RMP):

$$
\begin{array}{lccccccll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
 & \lambda_1, & & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, \quad \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941			

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1, 0, 0, 23, 0) = p_7$

Restricted master program (RMP):

$$
\begin{array}{lccccccccll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1, 0, 0, 23, 0) = p_7$

Restricted master program (RMP):

$$
\begin{array}{lccccccccll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & \geq & 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq & 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & \geq & 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & \geq & 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq & 899 & [\pi_5] \\
 & \lambda_1, & & \lambda_2, & & \lambda_3, & & \lambda_4, & \geq & 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1, 0, 0, 23, 0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941			

Restricted master program (RMP):

$$
\begin{array}{lccccccccll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & \geq & 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq & 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & \geq & 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & \geq & 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq & 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \geq & 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = p_8$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & & \lambda_3, & & \lambda_4, & & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = p_8$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & & \lambda_3, & & \lambda_4, & & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = p_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941			

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & & \lambda_3, & & \lambda_4, & & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = p_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941	1.0	0.0	

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{KP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = p_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = p_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = p_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941	1.0	0.0	Optimality!

Example (cont'd): The last RMP terminates with:

- Objective value $z_{LP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times

Example (cont'd): The last RMP terminates with:

- Objective value $z_{LP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times

Implications for Cutting Stock (CS) Problem, i.e., the integer program (IP):

- $LB = \lceil 127.451 \rceil = 128$ is a lower bound

Example (cont'd): The last RMP terminates with:

- Objective value $z_{LP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times 77
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times 22
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times 10
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times 17
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times 3

  Sum $\Sigma = 129$

Implications for Cutting Stock (CS) Problem, i.e., the integer program (IP):

- $LB = \lceil 127.451 \rceil = 128$ is a lower bound
- Rounding up the λ-values gives a feasible integer solution/upper bound $UB = 129$

Example (cont'd): The last RMP terminates with:

- Objective value $z_{LP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times 77
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times 22
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times 10
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times 17
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times 3

  Sum $\Sigma = 129$

Implications for Cutting Stock (CS) Problem, i.e., the integer program (IP):

- $LB = \lceil 127.451 \rceil = 128$ is a lower bound
- Rounding up the λ-values gives a feasible integer solution/upper bound $UB = 129$
- An optimal integer solution uses either 128 or 129 rolls

Example (cont'd): The last RMP terminates with:

- Objective value $z_{LP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times 77
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times 22
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times 10
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times 17
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times 3

  Sum $\Sigma = 129$

Implications for Cutting Stock (CS) Problem, i.e., the integer program (IP):

- $LB = \lceil 127.451 \rceil = 128$ is a lower bound
- Rounding up the λ-values gives a feasible integer solution/upper bound $UB = 129$
- An optimal integer solution uses either 128 or 129 rolls

Properties/conjectures:

- IRUP: $z(P) = \lceil z_{LP} \rceil$ Scheithauer and Terno (1992)
- MIRUP: $z(P) \leq \lceil z_{LP} \rceil + 1$ Scheithauer (2018, Sect. 4.11)

Branching: We typically do not branch on single λ-variables

Branching: We typically do not branch on single λ-variables

- Ryan-Foster branching for **BP**, i.e., for set-partitioning formulation
  - Items i_1 and i_2 are packed together into a bin:
    replace items by a new item of weight $w_1 + w_2$ → preserves structure
  - Items i_1 and i_2 must be are packed into different bins:
    add conflict constraint "$x_{i_1} + x_{i_2} \leq 1$" to KP → KPC → destroys structure

Branching: We typically do not branch on single λ-variables

- Ryan-Foster branching for **BP**, i.e., for set-partitioning formulation
  - Items i_1 and i_2 are packed together into a bin: replace items by a new item of weight $w_1 + w_2$ → preserves structure
  - Items i_1 and i_2 must be are packed into different bins: add conflict constraint "$x_{i_1} + x_{i_2} \leq 1$" to KP → KPC → destroys structure
- Vanderbeck's generic branching scheme applicable to **CS** → destroys structure (Vanderbeck, 1999, 2000)

Branching: We typically do not branch on single λ-variables

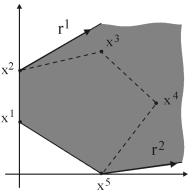
- Ryan-Foster branching for **BP**, i.e., for set-partitioning formulation
  - Items i_1 and i_2 are packed together into a bin: replace items by a new item of weight $w_1 + w_2$ → preserves structure
  - Items i_1 and i_2 must be are packed into different bins: add conflict constraint "$x_{i_1} + x_{i_2} \leq 1$" to KP → KPC → destroys structure
- Vanderbeck's generic branching scheme applicable to **CS** → destroys structure
  (Vanderbeck, 1999, 2000)

Cutting:

- Subset-row inequalities (SRIs) → destroys structure
  for set-partitioning formulation, i.e., **BP** (Wei *et al.*, 2020)
- Difficult for **CS**

Original formulation:

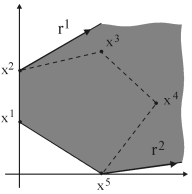
$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & Dx = d \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$



Original formulation:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & Dx = d \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

All elements $x \in X = \{Dx = d, x \in \mathbb{Z}^n_+\}$ can be expressed as



Original formulation:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & Dx = d \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

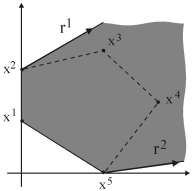
All elements $x \in X = \{Dx = d, x \in \mathbb{Z}^n_+\}$ can be expressed as

$$x = \sum_{p \in P} x^p \lambda_p + \sum_{r \in R} x^r \lambda_r$$

$$\text{with} \quad \sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0, p \in P; \quad \lambda_r \geq 0, r \in R$$

where $\{x^p : p \in P\}$ describes the set of all extreme points and $\{x^r : r \in R\}$ the set of all extreme rays of $conv(X)$.



Original formulation:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & Dx = d \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

Extensive formulation:

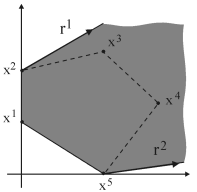
$$\begin{aligned} \min \quad & \sum_{p\in P} c_p^\top \lambda_p && + \sum_{r\in R} c_r^\top \lambda_r \\ \text{s.t.} \quad & \sum_{p} a_p \lambda_p && + \sum_{r\in R} a_r \lambda_r && = b \\ & \sum_{p\in P} \lambda_p = 1 \\ & \lambda_p \geq 0, p \in P, && \lambda_r \geq 0, r \in R \\ & x = \sum_{p\in P} x^p \lambda_p && + \sum_{r\in R} x^r \lambda_r && \in \mathbb{Z}^n_+ \end{aligned}$$

with $c_p = c^\top x^p$, $a_p = Ax^p$, $c_r = c^\top x^r$, and $a_r = Ax^r$.

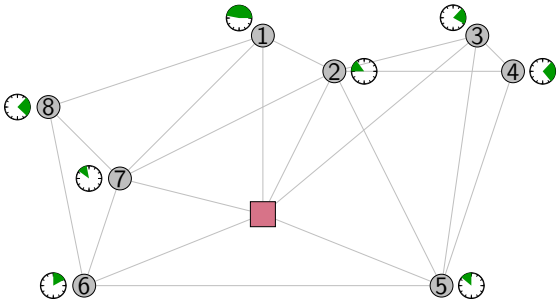
All elements $x \in X = \{Dx = d, x \in \mathbb{Z}^n_+\}$ can be expressed as

$$\begin{aligned} x = \quad & \sum_{p\in P} x^p \lambda_p && + \sum_{r\in R} x^r \lambda_r \\ \text{with} \quad & \sum_{p\in P} \lambda_p = 1 \\ & \lambda_p \geq 0, p \in P; \quad \lambda_r \geq 0, r \in R \end{aligned}$$

where $\{x^p : p \in P\}$ describes the set of all extreme points and $\{x^r : r \in R\}$ the set of all extreme rays of $conv(X)$.



Vehicle Routing Problem with Time Windows (VRPTW)



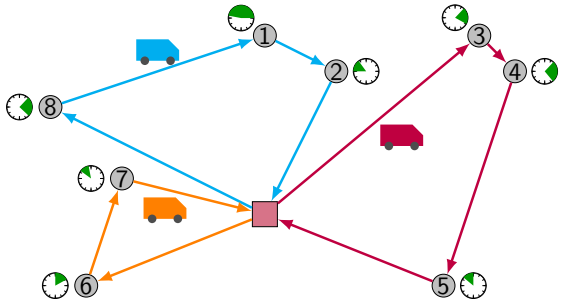
Given

- depot
- customers
- vehicles
- network

Problem specific

- time windows

Vehicle Routing Problem with Time Windows (VRPTW)



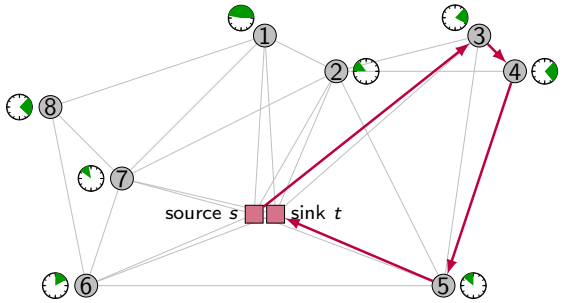
Given
- depot
- customers
- vehicles
- network

Problem specific
- time windows

Task: Find an optimal set of routes such that

- each customer is visited exactly once,
- each vehicle performs a single route,
- and all problem-specific constraints are satisfied.

Vehicle Routing Problem with Time Windows (VRPTW)



Given
- depot
- customers
- vehicles
- network

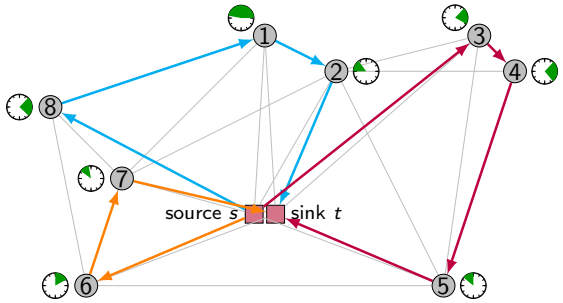
Problem specific
- time windows

Task: Find an optimal set of routes such that

- each customer is visited exactly once,
- each vehicle performs a single route,
- and all problem-specific constraints are satisfied.

Each vehicle route is an s-t-**path** in the underlying network!

Vehicle Routing Problem with Time Windows (VRPTW)



Given
- depot
- customers
- vehicles
- network

Problem specific
- time windows

Task: Find an optimal set of routes such that

- each customer is visited exactly once,
- each vehicle performs a single route,
- and all problem-specific constraints are satisfied.

Each vehicle route is an s-t-**path** in the underlying network!

Decision variables:

x_{ij}^k binary variable with $x_{ij}^k = 1$ if vehicle uses $k \in K$ arc $(i,j) \in A$, and $x_{ij}^k = 0$ otherwise

T_i^k continuous variable indicating the start of service of vehicle k at customer $i \in N$ if i is served by k

$$z_{3I-MTZ} = \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \quad (1)$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ij}^k = 1 \qquad i \in N \quad (2)$$

$$\sum_{(0,j) \in \delta^+(0)} x_{0j}^k = 1 \qquad k \in K \quad (3)$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji}^k = \sum_{(i,j) \in \delta^+(i)} x_{ij}^k \qquad i \in N, k \in K \quad (4)$$

$$\sum_{i \in N} q_i \sum_{(i,j) \in \delta^+(i)} x_{ij}^k \leq Q \qquad k \in K \quad (5)$$

$$T_i^k - T_j^k + M_{ij} x_{ij}^k \leq M_{ij} - t_{ij} \qquad (i,j) \in A, k \in K \quad (6)$$

$$a_i \leq T_i^k \leq b_i \qquad i \in N, k \in K \quad (7)$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A, k \in K$$

(1) minimize total routing costs

(2) each customer is served by exactly one vehicle

(3) each vehicle performs one route

(4) flow conservation

(5) capacity constraints

(6) setting the time variable T_i^k and elimination of subtours

(7) time window constraints

Decision variables:

x_{ij}^k binary variable with $x_{ij}^k = 1$ if vehicle uses $k \in K$ arc $(i,j) \in A$, and $x_{ij}^k = 0$ otherwise

T_i^k continuous variable indicating the start of service of vehicle k at customer $i \in N$ if i is served by k

$$z_{3I-MTZ} = \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \quad (1)$$

s.t.

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ij}^k = 1 \qquad i \in N \quad (2)$$

$$\sum_{(0,j) \in \delta^+(0)} x_{0j}^k = 1 \qquad k \in K \quad (3)$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji}^k = \sum_{(i,j) \in \delta^+(i)} x_{ij}^k \qquad i \in N, k \in K \quad (4)$$

$$\sum_{i \in N} q_i \sum_{(i,j) \in \delta^+(i)} x_{ij}^k \leq Q \qquad k \in K \quad (5)$$

$$T_i^k - T_j^k + M_{ij} x_{ij}^k \leq M_{ij} - t_{ij} \qquad (i,j) \in A, k \in K \quad (6)$$

$$a_i \leq T_i^k \leq b_i \qquad i \in N, k \in K \quad (7)$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A, k \in K$$

(1) minimize total routing costs

(2) each customer is served by exactly one vehicle

(3) each vehicle performs one route

(4) flow conservation

(5) capacity constraints

(6) setting the time variable T_i^k and elimination of subtours

(7) time window constraints

Properties of the formulation:

- Number of variables:
  $\rightarrow \mathcal{O}(|V|^2 \cdot |K|)$ binary variables
  $\rightarrow |N| \cdot |K|$ continuous variables
- $\mathcal{O}(|N^2| \cdot |K|)$ constraints

Properties of the formulation:

- Number of variables:
  $\rightarrow \mathcal{O}(|V|^2 \cdot |K|)$ binary variables
  $\rightarrow |N| \cdot |K|$ continuous variables
- $\mathcal{O}(|N^2| \cdot |K|)$ constraints
- Compact model
- Extensible in various ways

$\rightarrow$The Family of VRPs

Properties of the formulation:

- Number of variables:
  $\rightarrow \mathcal{O}(|V|^2 \cdot |K|)$ binary variables
  $\rightarrow |N| \cdot |K|$ continuous variables
- $\mathcal{O}(|N^2| \cdot |K|)$ constraints
- Compact model
- Extensible in various ways $\rightarrow$The Family of VRPs
- Weak linear relaxation
- Symmetry problem

Often block-diagonal structure:

$$D = \begin{pmatrix} D^1 & & & & \\ & D^2 & & & \\ & & D^3 & & \\ & & & \ddots & \\ & & & & D^K \end{pmatrix} \qquad x = \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ \vdots \\ x^K \end{pmatrix} \qquad d = \begin{pmatrix} d^1 \\ d^2 \\ d^3 \\ \vdots \\ d^K \end{pmatrix}$$

Often block-diagonal structure:

$$D = \begin{pmatrix} D^1 & & & & \\ & D^2 & & & \\ & & D^3 & & \\ & & & \ddots & \\ & & & & D^K \end{pmatrix} \qquad x = \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ \vdots \\ x^K \end{pmatrix} \qquad d = \begin{pmatrix} d^1 \\ d^2 \\ d^3 \\ \vdots \\ d^K \end{pmatrix}$$

Then $Dx = d$ decomposes into:

$$D^1 x^1 = d^1 \qquad D^2 x^2 = d^2 \qquad D^3 x^3 = d^3 \qquad \cdots \qquad D^{\bar{k}} x^{\bar{k}} = d^{\bar{k}}$$

Problem formulation with route variables for each vehicle k:

Definition of route variables:

Problem formulation with route variables for each vehicle k:

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} c_p \lambda_p^k$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} a_{ip} \lambda_p^k = 1 \qquad i \in N$$

$$\sum_{p \in P^k} \lambda_p^k \leq 1 \qquad k \in K$$

$$\lambda_p^k \in \{0,1\} \qquad k \in K, p \in P^k$$

Coefficients:

c_p cost of route p

a_{ip} $=1$ if route p visits customer i,
$=0$ otherwise

Variables:

λ_p^k $=1$ if route p of veh. k is part of the solution,
$=0$ otherwise

Definition of route variables:

Problem formulation with route variables for each vehicle k:

$$\min \quad \sum_{k\in K}\sum_{p\in P^k} c_p \lambda_p^k$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} a_{ip}\lambda_p^k = 1 \qquad i \in N$$

$$\sum_{p\in P^k} \lambda_p^k \le 1 \qquad k \in K$$

$$\lambda_p^k \in \{0,1\} \qquad k \in K, p \in P^k$$

Coefficients:

- c_p cost of route p
- a_{ip} $=1$ if route p visits customer i, $=0$ otherwise

Variables:

- λ_p^k $=1$ if route p of veh. k is part of the solution, $=0$ otherwise

Definition of route variables:

$$P^k = \left\{\text{extreme points } (x_{ij}^k, T_i^k)\right\}$$

of the set $X^k = \{D^k x^k, x^k \in \mathbb{Z}_+^{n_k}\}$ given by:

$$\sum_{(0,j)\in\delta^+(0)} x_{0j}^k = 1$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji}^k = \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \qquad i \in N$$

$$\sum_{i\in N} q_i \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \le Q$$

$$T_i^k - T_j^k + M_{ij}x_{ij}^k \le M_{ij} - t_{ij} \qquad (i,j) \in A(N)$$

$$a_i \le T_i^k \le b_i \qquad i \in N$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A$$

All vehicles are identical $\rightarrow$ Aggregation!

All vehicles are identical → Aggregation!

Aggregated formulation with route variables:

$$
\begin{aligned}
\min \quad & \sum_{p \in P} c_p \lambda_p & & \\
\text{s.t.} \quad & \sum_{p \in P} a_{ir} \lambda_p = 1 & & i \in N \\
& \sum_{p \in P} \lambda_p \leq |K| & & \\
& \lambda_p \in \{0,1\} & & p \in P
\end{aligned}
$$

All vehicles are identical → Aggregation!

Aggregated formulation with route variables:

$$\min \quad \sum_{p \in P} c_p \lambda_p$$

$$\text{s.t.} \sum_{p \in P} a_{ir} \lambda_p = 1 \qquad i \in N$$

$$\sum_{p \in P} \lambda_p \leq |K|$$

$$\lambda_p \in \{0, 1\} \qquad p \in P$$

⇒ **Path-based formulation**

All vehicles are identical → Aggregation!

Aggregated formulation with route variables:

$$\min \sum_{p \in P} c_p \lambda_p$$

$$\text{s.t.} \sum_{p \in P} a_{ir} \lambda_p = 1 \qquad i \in N$$

$$\sum_{p \in P} \lambda_p \leq |K|$$

$$\lambda_p \in \{0,1\} \qquad p \in P$$

⇒ **Path-based formulation**

Definition of aggregated route variables:

$$P = \{\text{extreme points } (x_{ij}, T_i)\}$$

of the set X given by:

$$\sum_{(0,j) \in \delta^+(0)} x_{0j} = 1$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = \sum_{(i,j) \in \delta^+(i)} x_{ij} \qquad i \in N$$

$$\sum_{i \in N} q_i y_i \leq Q$$

$$T_i - T_j + M_{ij} x_{ij} \leq M_{ij} - t_{ij} \qquad (i,j) \in A(N)$$

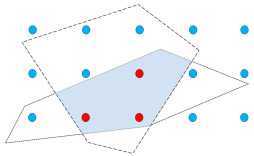
$$a_i \leq T_i \leq b_i \qquad i \in N$$

$$y_i \in \{0,1\} \qquad i \in N$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

LP relaxation original model:

$$\{Ax = b, x \geq 0\}$$
$$\cap \quad \{Dx = d, x \geq 0\}$$



Properties of the formulation:

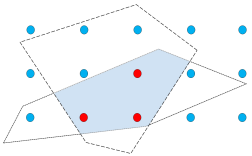
- Exponentially many variables
- $|N| + 1$ constraints
- Symmetry w.r.t. vehicle index k eliminated
- Stronger LP relaxation

Properties of the formulation:

- Exponentially many variables
- $|N| + 1$ constraints
- Symmetry w.r.t. vehicle index k eliminated
- Stronger LP relaxation

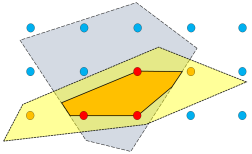
LP relaxation original model:

$$\{Ax = b, x \geq 0\}$$
$$\cap \quad \{Dx = d, x \geq 0\}$$



Reformulation, MP:

$$\{Ax = b, x \geq 0\}$$
$$\cap \quad conv(\{Dx = d, x \in \mathbb{Z}_+^n\})$$

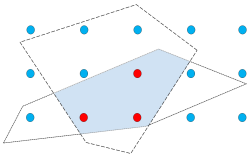


Properties of the formulation:

- Exponentially many variables
- $|N| + 1$ constraints
- Symmetry w.r.t. vehicle index k eliminated
- Stronger LP relaxation
- Solution by means of branch-price-and-cut (BPC):

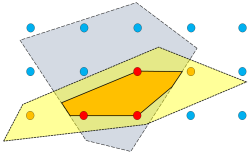
LP relaxation original model:

$$\{Ax = b, x \geq 0\}$$
$$\cap \quad \{Dx = d, x \geq 0\}$$



Reformulation, MP:

$$\{Ax = b, x \geq 0\}$$
$$\cap \quad conv(\{Dx = d, x \in \mathbb{Z}_+^n\})$$



Path-based formulations

Master program

$$\min \sum_{p \in P} c_p \lambda_p$$

$$\text{s.t.} \sum_{p \in P} a_{ip} \lambda_p = 1 \quad \forall i \in N$$

$$\lambda_p \in \{0, 1\} \quad \forall p \in P$$

- Variables $p \in P$ correspond to feasible routes
- Selects the best available routes
- Ensures that all customers are served

Problem: Number of feasible routes is large (often exponential)

Path-based formulations solved with column-generation techniques:

Restricted master program (RMP)

$$\min \sum_{p \in P'} c_p \lambda_p$$

$$\text{s.t.} \sum_{p \in P'} a_{ip} \lambda_p = 1 \quad \forall i \in N$$

$$\lambda_p \geq 0 \quad \forall p \in P'$$

- Variables $p \in P'$ correspond to feasible routes
- Selects the best available routes
- Ensures that all customers are served

Idea: Work with a (small) subset $P' \subset P$ of feasible routes

Path-based formulations solved with column-generation techniques:

Restricted master program (RMP)

$$\min \sum_{p \in P'} c_p \lambda_p$$

$$\text{s.t.} \sum_{p \in P'} a_{ip} \lambda_p = 1 \quad \forall i \in N$$

$$\lambda_p \geq 0 \quad \forall p \in P'$$

- Variables $p \in P'$ correspond to feasible routes
- Selects the best available routes
- Ensures that all customers are served

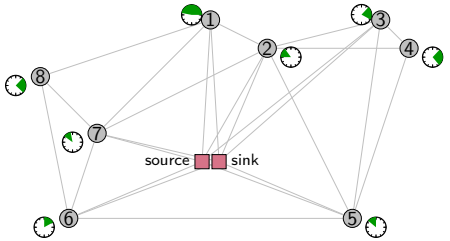
Idea: Work with a (small) subset $P' \subset P$ of feasible routes

Subproblem

- Identify missing routes:
  $\rightarrow$ Find a source-sink-path with negative reduced cost
- Shortest Path Problem with Resource Constraints

$$\tilde{c}_{ij} = c_{ij} - (\pi_i + \pi_j)/2$$

with $\pi_s = \pi_t := 0$



Path-based formulations solved with column-generation techniques:

Restricted master program (RMP)

$$\min \sum_{p \in P'} c_p \lambda_p$$

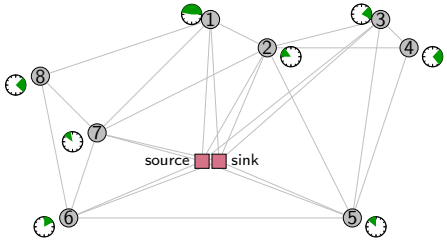
$$\text{s.t.} \sum_{p \in P'} a_{ip} \lambda_p = 1 \quad \forall i \in N$$

$$\lambda_p \geq 0 \quad \forall p \in P'$$

dual prices π_i for $i \in N$

Iterate!

negative reduced cost routes $p \in P \setminus P'$



Cutting:

- On variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation: "robust"

  Does not change feasible region of SP

  2-**path cuts** (Kohl *et al.*, 1999), for $S \subset N$ that requires at least 2 vehicles:

$$\sum_{p\in P} \sum_{(i,j)\in\delta^-(S)} b_{ij,p}\lambda_p \geq 2$$

Cutting:

- On variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation: "robust"

  Does not change feasible region of SP

  2-**path cuts** (Kohl *et al.*, 1999), for $S \subset N$ that requires at least 2 vehicles:

$$\sum_{p\in P} \sum_{(i,j)\in\delta^-(S)} b_{ij,p}\lambda_p \geq 2$$

- On variables λ of the master: "not robust"

  Subset-row inequalities (Jepsen *et al.*, 2008) for $S \subset N, |S| = 3$:

  changes and complicates SP

$$\sum_{p\in P} \left\lfloor \sum_{i\in S} a_{ip}/2 \right\rfloor \lambda_p \leq 1$$

Cutting:

- On variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation: "robust"

  Does not change feasible region of SP

  2-**path cuts** (Kohl *et al.*, 1999), for $S \subset N$ that requires at least 2 vehicles:

$$\sum_{p\in P} \sum_{(i,j)\in\delta^-(S)} b_{ij,p}\lambda_p \geq 2$$

- On variables λ of the master: "not robust"

  Subset-row inequalities (Jepsen *et al.*, 2008) for $S \subset N, |S| = 3$:

  changes and complicates SP

$$\sum_{p\in P} \left\lfloor \sum_{i\in S} a_{ip}/2 \right\rfloor \lambda_p \leq 1$$

  Non-robust capacity cuts (Baldacci *et al.*, 2008) for $S \subset N$ and $k(S)$ minimum number of vehicles needed to serve $S \subseteq N$:

$$\sum_{p\in P} \min \left\{ 1, \sum_{(i,j)\in\delta^-(S)} b_{ij,p} \right\} \lambda_p \geq k(S)$$

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

    Can change applicability of SP solver (e.g., DP)

    Branching on the schedule variables, i.e., $T_i \leq t$ or $T_i \geq t + 1$
    This introduces "new terms" in SP's objective, i.e., additional duals to be incorporated in SP

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

    Can change applicability of SP solver (e.g., DP)

    Branching on the schedule variables, i.e., $T_i \leq t$ or $T_i \geq t+1$
    This introduces "new terms" in SP's objective, i.e., additional duals to be incorporated in SP
  - Or impose constraint in x on SP

    Restrict feasible domain $\{x : Dx = d, x \in \mathbb{Z}^n_+\}$ of SP

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

    Can change applicability of SP solver (e.g., DP)

    Branching on the schedule variables, i.e., $T_i \leq t$ or $T_i \geq t+1$
    This introduces "new terms" in SP's objective, i.e., additional duals to be incorporated in SP
  - Or impose constraint in x on SP

    Restrict feasible domain $\{x : Dx = d, x \in \mathbb{Z}^n_+\}$ of SP

    Typically, branching on arcs with $x_{ij} \in \{0,1\}$

Branching:

- Branching on variables $x = (x_{ij})_{(i,j)\in A}$ of the original formulation:
  We can guarantee integrality due to $x = \sum_p a_{ip}\lambda_p + \sum_r a_{ir}\lambda_r \in \mathbb{Z}^n_+$
  - Either impose constraint in x on MP

    Does not change feasible region of SP

    Branching on the number of vehicles, i.e., $\sum_{(0,j)\in A} x_{0j} \in \mathbb{Z}_+$

    Can change applicability of SP solver (e.g., DP)

    Branching on the schedule variables, i.e., $T_i \leq t$ or $T_i \geq t+1$
    This introduces "new terms" in SP's objective, i.e., additional duals to be incorporated in SP
  - Or impose constraint in x on SP

    Restrict feasible domain $\{x : Dx = d, x \in \mathbb{Z}^n_+\}$ of SP

    Typically, branching on arcs with $x_{ij} \in \{0,1\}$
- Branching on variables λ of the master:
  Typically, we do not do this!

The Family of Vehicle Routing Problems

- Pickup and Delivery with TW (Ropke and Cordeau, 2009)
- LIFO Pickup and Delivery (Cherkesly *et al.*, 2015)
- Simultaneous Delivery and Pickup (Halse, 1992)
- Ride-time constraints in DARP (Gschwind and Irnich, 2015), (Gschwind, 2015)
- Split Delivery VRPTW (Desaulniers, 2010)
- Hours of service regulations (Prescott-Gagnon *et al.*, 2010), (Goel and Irnich, 2016)
- Time-dependent VRPTW (Dabia *et al.*, 2013)
- Truck-and-trailer (Rothenbächer *et al.*, 2017)
- Traveling Purchaser with unitary demand (Bianchessi *et al.*, 2021)
- Fragility-Constrained VRPTW (Altman *et al.*, 2023)
- Electric VRPTW (Desaulniers *et al.*, 2016)
  - non-linear (Montoya *et al.*, 2017)
  - non-linear (Lam *et al.*, 2022)
  - ...
- Robust VRPTW (Munari *et al.*, 2019)
- Stochastic VRPs
  - stochastic demands (Florio *et al.*, 2020)
  - stochastic and correlated travel times (Rostami *et al.*, 2021)
  - stochastic demands and partial reoptimization (Florio *et al.*, 2022)
  - ...
- ... (many, many more!)

Why do we use Branch-Price-and-Cut Algorithms?

1. The extensive formulation often has a stronger linear-relaxation bound compared to the original formulation;
2. an extensive formulation can eliminate inherent symmetry;
3. column-generation subproblems can cope with non-linearities of the original formulation; and
4. powerful algorithms have been developed to solve subproblems for important real-world and theoretical problems.

Basic Components:

- column generation (=pricing)
- branching
- cutting

Basic Components:

- column generation (=pricing)
- branching
- cutting

Advanced Components

At subproblem level:

- pricing problem relaxation
- heuristic pricing
- variable fixing by reduced costs

Basic Components:

- column generation (=pricing)
- branching
- cutting

Advanced Components

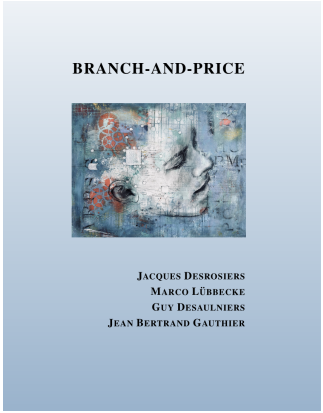
At subproblem level:

- pricing problem relaxation
- heuristic pricing
- variable fixing by reduced costs

At master program level:

- stabilization
- column enumeration + MIP solver
- strong branching
- primal heuristics

New book on BPC by Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier (Desrosiers *et al.*, 2024):
`https://doi.org/10.13140/RG.2.2.29888.14088`



More helpful material in habilitation thesis of Ruslan Sadykov (2019).

Part: The Shortest Path Problem with Resource Constraints (SPPRC)

What is the SPPRC?

For vehicle and crew scheduling and routing problems (airlines, public transport, rail, shipping, and logistics service providers), the pricing problem is a variant of shortest path problem with resource constraints (SPPRC).

What is the SPPRC?

For vehicle and crew scheduling and routing problems (airlines, public transport, rail, shipping, and logistics service providers), the pricing problem is a variant of shortest path problem with resource constraints (SPPRC).

Time-Space Networks

- vehicle and crew scheduling
- acyclic or "few, rather long" (task) cycles
- many resources
- simple resource propagation (often additive)

(Himmich *et al.*, 2024)

What is the SPPRC?

For vehicle and crew scheduling and routing problems (airlines, public transport, rail, shipping, and logistics service providers), the pricing problem is a variant of shortest path problem with resource constraints (SPPRC).

Time-Space Networks

- vehicle and crew scheduling
- acyclic or "few, rather long" (task) cycles
- many resources
- simple resource propagation (often additive)

Spacial Networks

- vehicle routing
- "many" cycles ($i - j - i$ possible)
- few(er) resources
- can have non-linear, highly complex resource extension functions (REFs)

(Himmich *et al.*, 2024)

What is the SPPRC?

For vehicle and crew scheduling and routing problems (airlines, public transport, rail, shipping, and logistics service providers), the pricing problem is a variant of shortest path problem with resource constraints (SPPRC).

Time-Space Networks

- vehicle and crew scheduling
- acyclic or "few, rather long" (task) cycles
- many resources
- simple resource propagation (often additive)

Spacial Networks

- vehicle routing
- "many" cycles ($i - j - i$ possible)
- few(er) resources
- can have non-linear, highly complex resource extension functions (REFs)

(Himmich *et al.*, 2024)

Airlines, Rail, and Public Transport

- Vehicle Scheduling (incl. maintenance)
- Crew Scheduling, Rostering
- Integrated Problems

Vehicle Routing Problems

What is the SPPRC?

Given: Digraph $D = (V, A)$ with

- source/start/origin $s \in V$ and sink/end/destination $t \in V$
- (reduced) cost $\tilde{c}_{ij}$ for all $(i, j) \in A$
- resource constraints defines by REFs $f_{ij} : \mathbb{Q}^r \to \mathbb{Q}^r$ for all arcs $(i, j) \in A$ and resource intervals $[L_i, U_i]$ for all vertices $i \in V$

Find: Minimum (reduced) cost s-to-t-path respecting resource constraints

What is the SPPRC?

Given: Digraph $D = (V, A)$ with

- source/start/origin $s \in V$ and sink/end/destination $t \in V$
- (reduced) cost $\tilde{c}_{ij}$ for all $(i,j) \in A$
- resource constraints defines by REFs $f_{ij} : \mathbb{Q}^r \to \mathbb{Q}^r$ for all arcs $(i,j) \in A$ and resource intervals $[L_i, U_i]$ for all vertices $i \in V$

Find: Minimum (reduced) cost s-to-t-path respecting resource constraints

Model for acyclic digraphs D:

$$\min \quad \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(j,i)\in\delta^-(i)} x_{ji} = \begin{cases} +1, & i = s \\ 0, & i \in V, i \neq s, t \\ -1, & i = t \end{cases}$$

$$x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) \qquad (i,j) \in A$$

$$L_i \leq F_i \leq U_i, \qquad i \in V$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Given: Digraph $D = (V, A)$ with

- source/start/origin $s \in V$ and sink/end/destination $t \in V$
- (reduced) cost $\tilde{c}_{ij}$ for all $(i, j) \in A$
- resource constraints defines by REFs $f_{ij} : \mathbb{Q}^r \to \mathbb{Q}^r$ for all arcs $(i, j) \in A$ and resource intervals $[L_i, U_i]$ for all vertices $i \in V$

Find: Minimum (reduced) cost s-to-t-path respecting resource constraints

Model for acyclic digraphs D:

$$\min \quad \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(j,i)\in\delta^-(i)} x_{ji} = \begin{cases} +1, & i = s \\ 0, & i \in V, i \neq s, t \\ -1, & i = t \end{cases}$$

$$x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) \qquad (i,j) \in A$$

$$L_i \leq F_i \leq U_i, \qquad i \in V$$

$$x_{ij} \in \{0, 1\} \qquad (i,j) \in A$$

Typically not solved as a MIP but with a dynamic programming labeling algorithm.

What is the SPPRC?

Given: Digraph $D = (V, A)$ with

- source/start/origin $s \in V$ and sink/end/destination $t \in V$
- (reduced) cost $\tilde{c}_{ij}$ for all $(i, j) \in A$
- resource constraints defines by REFs $f_{ij} : \mathbb{Q}^r \to \mathbb{Q}^r$ for all arcs $(i, j) \in A$ and resource intervals $[L_i, U_i]$ for all vertices $i \in V$

Find: Minimum (reduced) cost s-to-t-path respecting resource constraints

Model for acyclic digraphs D: w/o path structural constraints, e.g., elementarity

$$\min \quad \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(j,i)\in\delta^-(i)} x_{ji} = \begin{cases} +1, & i = s \\ 0, & i \in V, i \neq s, t \\ -1, & i = t \end{cases}$$

$$x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) \qquad (i,j) \in A$$

$$L_i \leq F_i \leq U_i, \qquad i \in V$$

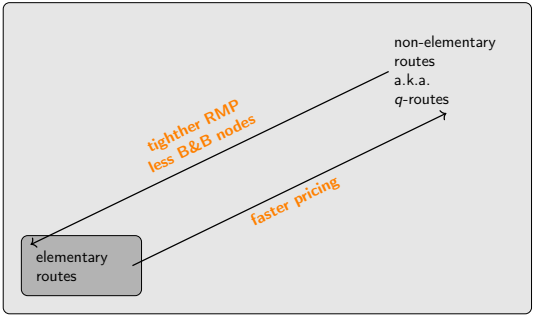
$$x_{ij} \in \{0, 1\} \qquad (i,j) \in A$$

Typically not solved as a MIP but with a dynamic programming labeling algorithm.

Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

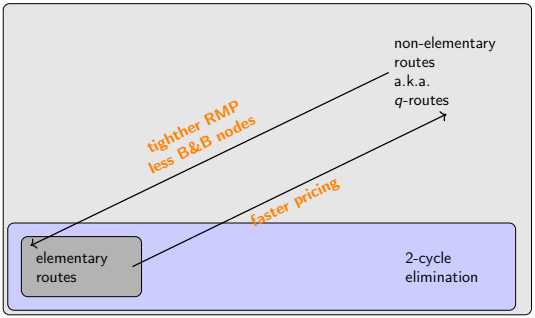
- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound



Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

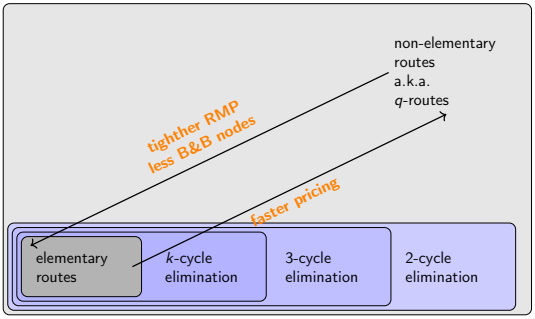
- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound
- 2-cycle free (Houck *et al.*, 1980)

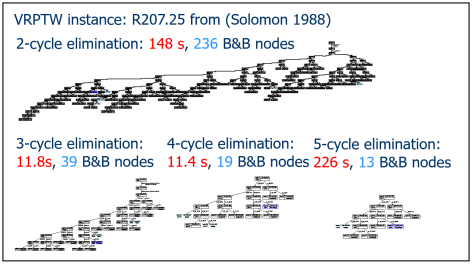


Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound
- 2-cycle free (Houck *et al.*, 1980)
- k-cycle (Irnich and Villeneuve, 2006)

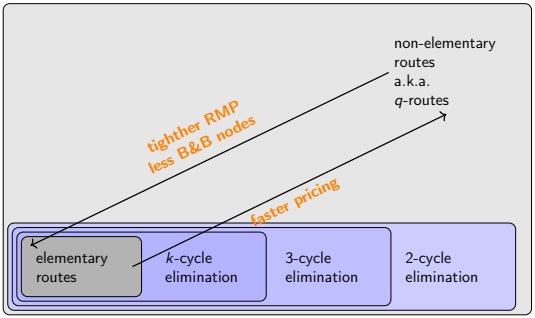




Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

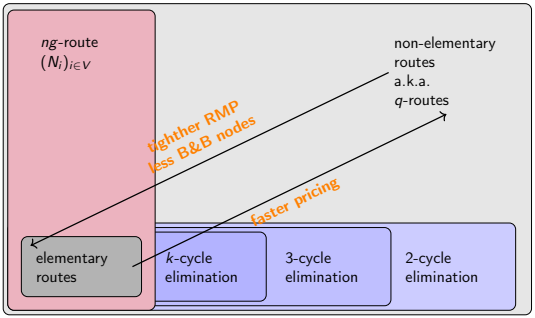
- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound
- 2-cycle free (Houck *et al.*, 1980)
- k-cycle (Irnich and Villeneuve, 2006)
- Partial elementarity (Desaulniers *et al.*, 2008)



Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound
- 2-cycle free (Houck *et al.*, 1980)
- k-cycle (Irnich and Villeneuve, 2006)
- Partial elementarity (Desaulniers *et al.*, 2008)

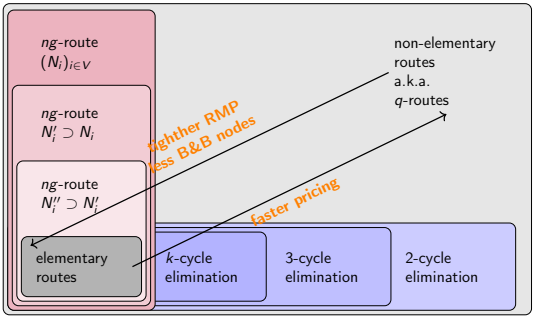


- *ng*-route (Baldacci *et al.*, 2011)
  - Define for each node $i \in V$ a neighborhood $N_i \subset V \setminus \{s, t\}$
  - A cycle $(i, \ldots, j, \ldots, i)$ is only allowed (=feasible), if there is a node j in the cycle with $i \notin N_j$

Elementary SPPRC (ESPPRC) is NP-hard in strong sense (Dror, 1994), (Spliet, 2023). Intense research on dynamic programming labeling algorithm for (E)SPPRC:

Relaxations:

- non-elementary routes a.k.a. q-routes (Christofides *et al.*, 1981)
  $\rightarrow$additional col. with coeff. $a_{ip} \geq 2$, never in integer solution, deteriorate MP bound
- 2-cycle free (Houck *et al.*, 1980)
- k-cycle (Irnich and Villeneuve, 2006)
- Partial elementarity (Desaulniers *et al.*, 2008)



- *ng*-route (Baldacci *et al.*, 2011)
  - Define for each node $i \in V$ a neighborhood $N_i \subset V \setminus \{s, t\}$
  - A cycle $(i, \ldots, j, \ldots, i)$ is only allowed (=feasible), if there is a node j in the cycle with $i \notin N_j$
  - Fine control of tradeoff between strength of the LP-relaxation (RMP) and difficulty of the SP

Different approaches to reach the elementary lower bound are discussed in (Contardo *et al.*, 2015):

- Decremetal State Space Relaxation (DSSR) (Boland *et al.*, 2006), (Righini and Salani, 2008)
  $\rightarrow$ enforce elementarity only via subproblem

Different approaches to reach the elementary lower bound are discussed in (Contardo *et al.*, 2015):

- Decremetal State Space Relaxation (DSSR) (Boland *et al.*, 2006), (Righini and Salani, 2008)
  → enforce elementarity only via subproblem
- Strong degree cuts (Contardo *et al.*, 2014)
  → enforce elementarity only via constraints in RMP
  → non-robust cut must be handled in SPPRC subproblem

More acceleration techniques:

- Heuristics (cascade of heuristics per iteration; partial pricing (Gamache *et al.*, 1999)—generate feasible, negative rdc paths, not necessarily with minimum rdc)
  - Primal heuristics (local search, tabu search, add and drop, LNS etc.) (Desaulniers *et al.*, 2008), (Ropke and Cordeau, 2009), (Parragh *et al.*, 2012), ...
  - Labeling-based heuristics:
    - smaller networks (thinned out w.r.t. rdc/proximity) (Dumas *et al.*, 1991)
    - re-use of labels (Feillet *et al.*, 2007)
    - stronger dominance, e.g., on proper subset of resources
    - limited discrepancy search (Feillet *et al.*, 2007)
- Bidirectional labeling (Righini and Salani, 2006), (Tilk *et al.*, 2017)
- Completion Bounds (Contardo and Martinelli, 2014), (Bode and Irnich, 2015)
- Selective Pricing (Desaulniers *et al.*, 2017)
- Arc-based memory for *ng*-route relaxation (Pecin *et al.*, 2017b), (Bulhões *et al.*, 2018)
- Limited-memory for SRIs (Pecin *et al.*, 2017a)
- Improved dominance algorithms: buckets (Sadykov *et al.*, 2021), partial dominance (Ioachim *et al.*, 1998)
- Arc Elimination/Arc fixing (Irnich *et al.*, 2010), (Desaulniers *et al.*, 2020)

More acceleration techniques:

- Heuristics (cascade of heuristics per iteration; partial pricing (Gamache *et al.*, 1999)—generate feasible, negative rdc paths, not necessarily with minimum rdc)
  - Primal heuristics (local search, tabu search, add and drop, LNS etc.) (Desaulniers *et al.*, 2008), (Ropke and Cordeau, 2009), (Parragh *et al.*, 2012), ...
  - Labeling-based heuristics:
    - smaller networks (thinned out w.r.t. rdc/proximity) (Dumas *et al.*, 1991)
    - re-use of labels (Feillet *et al.*, 2007)
    - stronger dominance, e.g., on proper subset of resources
    - limited discrepancy search (Feillet *et al.*, 2007)
- Bidirectional labeling (Righini and Salani, 2006), (Tilk *et al.*, 2017)
- Completion Bounds (Contardo and Martinelli, 2014), (Bode and Irnich, 2015)
- Selective Pricing (Desaulniers *et al.*, 2017)
- Arc-based memory for *ng*-route relaxation (Pecin *et al.*, 2017b), (Bulhões *et al.*, 2018)
- Limited-memory for SRIs (Pecin *et al.*, 2017a)
- Improved dominance algorithms: buckets (Sadykov *et al.*, 2021), partial dominance (Ioachim *et al.*, 1998)
- Arc Elimination/Arc fixing (Irnich *et al.*, 2010), (Desaulniers *et al.*, 2020)

The SPPRC with forward REFs f_{ij}:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \mathcal{N}\mathrm{x} = \mathrm{u}_s - \mathrm{u}_t \\
& x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) && (i,j) \in A \\
& L_i \leq F_i \leq U_i, && i \in V \\
& x_{ij} \in \{0,1\} && (i,j) \in A
\end{aligned}
$$

with incidence matrix $\mathcal{N}$ of $G = (V, A)$ and unit vectors $\mathrm{u}_s, \mathrm{u}_t$.

The SPPRC with forward REFs f_{ij}:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \mathcal{N}\mathrm{x} = \mathrm{u}_s - \mathrm{u}_t \\
& x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) && (i,j) \in A \\
& L_i \leq F_i \leq U_i, && i \in V \\
& x_{ij} \in \{0,1\} && (i,j) \in A
\end{aligned}
$$

with incidence matrix $\mathcal{N}$ of $G = (V, A)$ and unit vectors $\mathrm{u}_s, \mathrm{u}_t$.

The SPPRC with backward REFs b_{ij}:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \mathcal{N}\mathrm{x} = \mathrm{u}_s - \mathrm{u}_t \\
& x_{ij} = 1 \quad \Rightarrow \quad B_i \leq b_{ij}(B_j) && (i,j) \in A \\
& L_i \leq B_i \leq U_i, && i \in V \\
& x_{ij} \in \{0,1\} && (i,j) \in A
\end{aligned}
$$

Bidirectional Labeling

The SPPRC with forward REFs f_{ij}:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \mathcal{N}\mathrm{x} = \mathrm{u}_s - \mathrm{u}_t \\
& x_{ij} = 1 \quad \Rightarrow \quad F_j \geq f_{ij}(F_i) \qquad (i,j) \in A \\
& L_i \leq F_i \leq U_i, \qquad i \in V \\
& x_{ij} \in \{0,1\} \qquad (i,j) \in A
\end{aligned}
$$

The SPPRC with backward REFs b_{ij}:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \mathcal{N}\mathrm{x} = \mathrm{u}_s - \mathrm{u}_t \\
& x_{ij} = 1 \quad \Rightarrow \quad B_i \leq b_{ij}(B_j) \qquad (i,j) \in A \\
& L_i \leq B_i \leq U_i, \qquad i \in V \\
& x_{ij} \in \{0,1\} \qquad (i,j) \in A
\end{aligned}
$$

with incidence matrix $\mathcal{N}$ of $G = (V, A)$ and unit vectors $\mathrm{u}_s, \mathrm{u}_t$.

Desirable properties (Irnich, 2008):

$$
\begin{array}{rcl}
f_{ij}(F_i) \leq F_j & \iff & B_i \leq b_{ij}(B_j) \\
F_i \leq F_i' \quad \Rightarrow \quad f_{ij}(F_i) \leq f_{ij}(F_i') & & B_j \leq B_j' \quad \Rightarrow \quad b_{ij}(B_j) \leq b_{ij}(B_j')
\end{array}
$$

i.e., consistency (identical feasible paths) and non-decreasing.

Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q

  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$

  load residual capacity

Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q
  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$
  load residual capacity
- Time windows $[e_i, \ell_i]$
  $f_{ij}(F_i) = \max\{e_j, F_i + t_{ij}\} \leq \ell_j$ and $b_{ij}(B_j) = \min\{\ell_i, B_j - t_{ij}\} \geq e_i$
  earliest time latest time

Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q
  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$
  load residual capacity
- Time windows $[e_i, \ell_i]$
  $f_{ij}(F_i) = \max\{e_j, F_i + t_{ij}\} \leq \ell_j$ and $b_{ij}(B_j) = \min\{\ell_i, B_j - t_{ij}\} \geq e_i$
  earliest time latest time
- Two interdependent resources, e.g., as in VRP with simultaneous delivery and pickup
  $f_{ij}(F_i^1, F_i^2) = (\max\{F_i^1 + t_{ij}, F_i^2 + s_{ij}\}, \max\{F_i^1 + t'_{ij}, F_i^2 + s'_{ij}\})$
  and $b_{ij}(B_j^1, B_j^2) = (\min\{B_j^1 - t_{ij}, B_j^2 - t'_{ij}\}, \min\{B_j^1 - s_{ij}, B_j^2 - s'_{ij}\})$
  accumulated pickups max load
  max load residual cap. w.r.t. deliveries

Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q
  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$
  load residual capacity
- Time windows $[e_i, \ell_i]$
  $f_{ij}(F_i) = \max\{e_j, F_i + t_{ij}\} \leq \ell_j$ and $b_{ij}(B_j) = \min\{\ell_i, B_j - t_{ij}\} \geq e_i$
  earliest time latest time
- Two interdependent resources, e.g., as in VRP with simultaneous delivery and pickup
  $f_{ij}(F_i^1, F_i^2) = (\max\{F_i^1 + t_{ij}, F_i^2 + s_{ij}\}, \max\{F_i^1 + t'_{ij}, F_i^2 + s'_{ij}\})$
  and $b_{ij}(B_j^1, B_j^2) = (\min\{B_j^1 - t_{ij}, B_j^2 - t'_{ij}\}, \min\{B_j^1 - s_{ij}, B_j^2 - s'_{ij}\})$
  accumulated pickups max load
  max load residual cap. w.r.t. deliveries
- Likewise for TWs and minimum tour duration

Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q
  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$
  load residual capacity
- Time windows $[e_i, \ell_i]$
  $f_{ij}(F_i) = \max\{e_j, F_i + t_{ij}\} \leq \ell_j$ and $b_{ij}(B_j) = \min\{\ell_i, B_j - t_{ij}\} \geq e_i$
  earliest time latest time
- Two interdependent resources, e.g., as in VRP with simultaneous delivery and pickup
  $f_{ij}(F_i^1, F_i^2) = (\max\{F_i^1 + t_{ij}, F_i^2 + s_{ij}\}, \max\{F_i^1 + t'_{ij}, F_i^2 + s'_{ij}\})$
  and $b_{ij}(B_j^1, B_j^2) = (\min\{B_j^1 - t_{ij}, B_j^2 - t'_{ij}\}, \min\{B_j^1 - s_{ij}, B_j^2 - s'_{ij}\})$
  accumulated pickups max load
  max load residual cap. w.r.t. deliveries
- Likewise for TWs and minimum tour duration
- Arc-specific resource windows
  $f_{ij}(F_i) = \max\{e_{ij}, F_i + t_{ij}\}$ and $b_{ij}(B_j) = \min\{\ell_{ij}, B_j - t_{ij}\}$

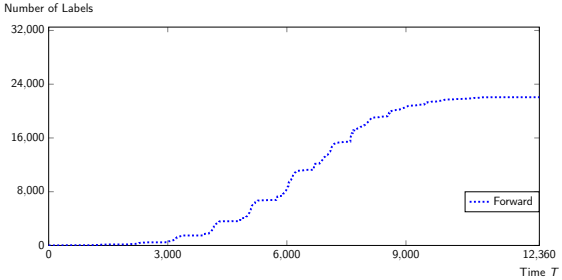
Bidirectional Labeling

Standard technique for many types of REFs such as

- Vehicle capacity Q
  $f_{ij}(F_i) = F_i + d_j \leq Q$ and $b_{ij}(B_j) = B_j - d_j \geq d_i$
  load residual capacity
- Time windows $[e_i, \ell_i]$
  $f_{ij}(F_i) = \max\{e_j, F_i + t_{ij}\} \leq \ell_j$ and $b_{ij}(B_j) = \min\{\ell_i, B_j - t_{ij}\} \geq e_i$
  earliest time latest time
- Two interdependent resources, e.g., as in VRP with simultaneous delivery and pickup
  $f_{ij}(F_i^1, F_i^2) = (\max\{F_i^1 + t_{ij}, F_i^2 + s_{ij}\}, \max\{F_i^1 + t'_{ij}, F_i^2 + s'_{ij}\})$
  and $b_{ij}(B_j^1, B_j^2) = (\min\{B_j^1 - t_{ij}, B_j^2 - t'_{ij}\}, \min\{B_j^1 - s_{ij}, B_j^2 - s'_{ij}\})$
  accumulated pickups max load
  max load residual cap. w.r.t. deliveries
- Likewise for TWs and minimum tour duration
- Arc-specific resource windows
  $f_{ij}(F_i) = \max\{e_{ij}, F_i + t_{ij}\}$ and $b_{ij}(B_j) = \min\{\ell_{ij}, B_j - t_{ij}\}$
- Inversion is not always straightforward!
  $\rightarrow$ EVRPTW with full recharges (Desaulniers *et al.*, 2016)

Labeling algorithms suffer from **combinatorial explosion**:

Example:
VRPTW with time horizon $[0, 12\,360]$



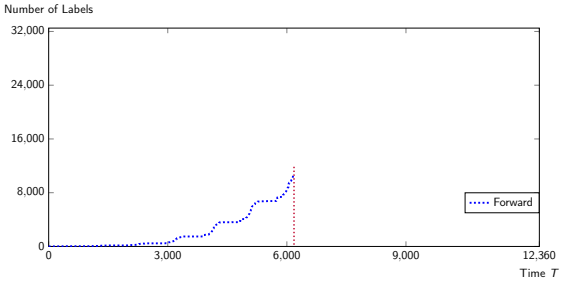
Main idea: *'Symmetry helps'* (Righini and Salani, 2006)
$\rightarrow$ Avoid long path by using bidirectional labeling:

1. Extend labels from the source in forward direction

Bidirectional Labeling

Labeling algorithms suffer from **combinatorial explosion**:

Example:
VRPTW with time horizon $[0, 12\,360]$



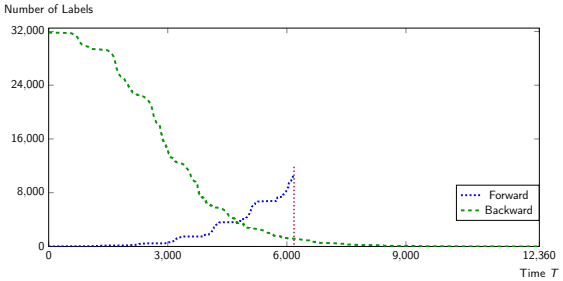
Main idea: *'Symmetry helps'* (Righini and Salani, 2006)
$\rightarrow$ Avoid long path by using bidirectional labeling:

1. Extend labels from the source in forward direction only up to 'the middle' (half-way point)

Bidirectional Labeling

Labeling algorithms suffer from **combinatorial explosion**:

Example:
VRPTW with time horizon $[0, 12\,360]$



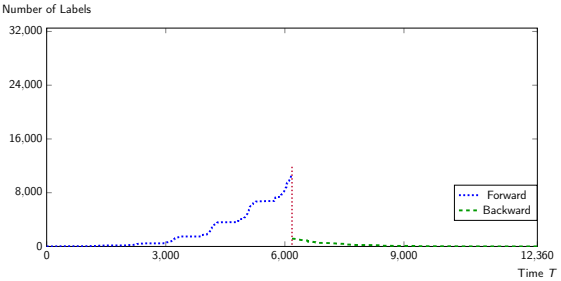
Main idea: *'Symmetry helps'* (Righini and Salani, 2006)
$\rightarrow$ Avoid long path by using bidirectional labeling:

1. Extend labels from the source in forward direction only up to 'the middle' (half-way point)
2. Extend labels from the sink in backward direction

Bidirectional Labeling

Labeling algorithms suffer from **combinatorial explosion**:

Example:
VRPTW with time horizon $[0, 12\,360]$



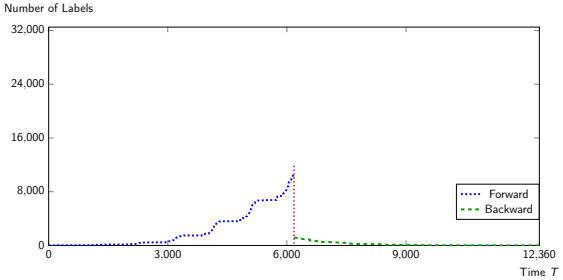
Main idea: *'Symmetry helps'* (Righini and Salani, 2006)
$\rightarrow$ Avoid long path by using bidirectional labeling:

1. Extend labels from the source in forward direction only up to 'the middle' (half-way point)
2. Extend labels from the sink in backward direction only up to 'the middle' (half-way point)

Bidirectional Labeling

Labeling algorithms suffer from **combinatorial explosion**:

Example:
VRPTW with time horizon $[0, 12\,360]$



Main idea: *'Symmetry helps'* (Righini and Salani, 2006)
$\rightarrow$ Avoid long path by using bidirectional labeling:

1. Extend labels from the source in forward direction only up to 'the middle' (half-way point)
2. Extend labels from the sink in backward direction only up to 'the middle' (half-way point)
3. Merge suitable forward and backward labels to obtain complete paths

Refinement: Choose the half-way point dynamically (Tilk *et al.*, 2017)

Algorithm 1: Bi-Directional Labeling Algorithm with Dynamic Half-Way Points

Input: ..., forward and backward half-way points H_F and H_B (required $H_F \geq H_B$)

```
1  F := F(o), B := B(d);
2  while F ≠ null or B ≠ null do
3      direction := getNextDirection();
4      if direction = forward then
5          if F^res ≤ H_F then
6              for (i, j) ∈ A with i = v(F) do
7                  Propagate F to vertex j with REF f_ij
8          H_B := max{H_B, min{F^res, H_F}};
9          F := getNextForwardLabel();
10     else
11         if B^res > H_B then
12             for (i, j) ∈ A with j = v(B) do
13                 Propagate B to vertex i with REF b_ij
14         H_F := min{H_F, max{B^res, H_B}};
15         B := getNextBackwardLabel();
16     if callDominanceAlgorithm() then
17         Apply dominance rules;
18 Merge forward labels F and backward labels B with v(F) = v(B);
```

Algorithm 1: Bi-Directional Labeling Algorithm with Dynamic Half-Way Points

Input: ..., forward and backward half-way points H_F and H_B (required $H_F \geq H_B$)

```
1  F := F(o), B := B(d);
2  while F ≠ null or B ≠ null do
3      direction := getNextDirection();
4      if direction = forward then
5          if F^res ≤ H_F then
6              for (i, j) ∈ A with i = v(F) do
7                  Propagate F to vertex j with REF f_ij
8          H_B := max{H_B, min{F^res, H_F}};
9          F := getNextForwardLabel();
10     else
11         if B^res > H_B then
12             for (i, j) ∈ A with j = v(B) do
13                 Propagate B to vertex i with REF b_ij
14         H_F := min{H_F, max{B^res, H_B}};
15         B := getNextBackwardLabel();
16     if callDominanceAlgorithm() then
17         Apply dominance rules;
18 Merge forward labels F and backward labels B with v(F) = v(B);
```

$H_F = U$, $H_B = L$: full flexibility
$H_F = H_B = H$: Righini & Salani
$H_F = H_B = U$: forward labeling
$H_F = H_B = L$: backward labeling

Function getNextDirection() should return

forward if estimated effort of forward labeling is smaller than effort of backward labeling;

backward if estimated oppositely.

Function getNextDirection() should return

forward if estimated effort of forward labeling is smaller than effort of backward labeling;

backward if estimated oppositely.

Tested: Function getNextDirection()

1. Choose direction with the smaller number of generated labels
2. Choose direction with the smaller number of processed labels
3. Choose direction with the smaller number of unprocessed labels

Bidirectional Labeling

Function getNextDirection() should return

forward if estimated effort of forward labeling is smaller than effort of backward labeling;

backward if estimated oppositely.

Tested: Function getNextDirection()

1. Choose direction with the smaller number of generated labels
2. Choose direction with the smaller number of processed labels
3. Choose direction with the smaller number of unprocessed labels

Result: Last strategy based on number of unprocessed labels often works best!

Part: Reduced-Cost based Variable Elimination/Fixing

Proposition 1 (Nemhauser and Wolsey (1988), Proposition 2.1, page 389)

Let UB be an upper bound on the optimal value of the minimization problem M, and let π be a dual solution to the linear relaxation of M providing a lower bound $LB(\pi)$.
If an integer variable $x \geq 0$ has reduced cost

$$\tilde{c}_x(\pi) > UB - LB(\pi),$$

then $x = 0$ in every optimal solution to M, i.e., x can be eliminated.

Proposition 1 (Nemhauser and Wolsey (1988), Proposition 2.1, page 389)

Let UB be an upper bound on the optimal value of the minimization problem M, and let π be a dual solution to the linear relaxation of M providing a lower bound $LB(\pi)$.
If an integer variable $x \geq 0$ has reduced cost

$$\tilde{c}_x(\pi) > UB - LB(\pi),$$

then $x = 0$ in every optimal solution to M, i.e., x can be eliminated.

Not directly applicable in column generation:

- Forbidding the re-generation of one or several variables changes the structure of the pricing subproblem
- Possible via network modification (Villeneuve and Desaulniers, 2005); effort of solving modified subproblems is often too high

$$
\begin{aligned}
\min \quad & \sum_{p \in P} c_p \lambda_p & & \\
\text{subject to} \quad & \sum_{p \in P} a_{kp} \lambda_p = 1 & \forall k \in K & \quad [\pi] \\
& \lambda_p \geq 0 \quad \text{integer} & \forall p \in P &
\end{aligned}
$$

with

K: set of tasks to fulfill

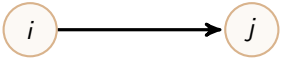
P: set of all resource feasible paths, underlying network $D = (V, A)$

MP: linear relaxation, i.e., $\lambda_p \geq 0$, $\lambda_p \in \mathbb{Q}$

RMP: linear relaxation defined of subset $P' \subset P$

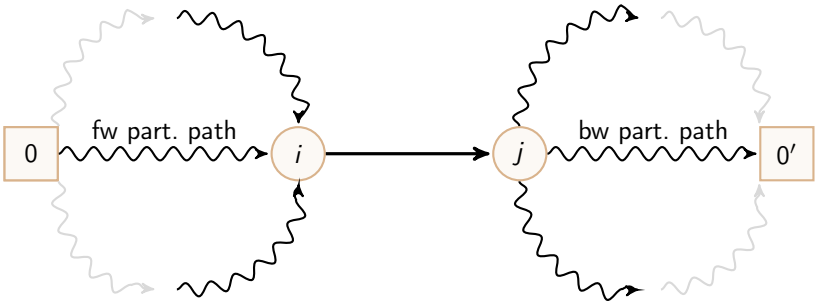
π: dual solution of MP/RMP

Consider an arc $(i, j) \in A$ and the set $P[ij]$ of all paths that contain the arc (i, j).

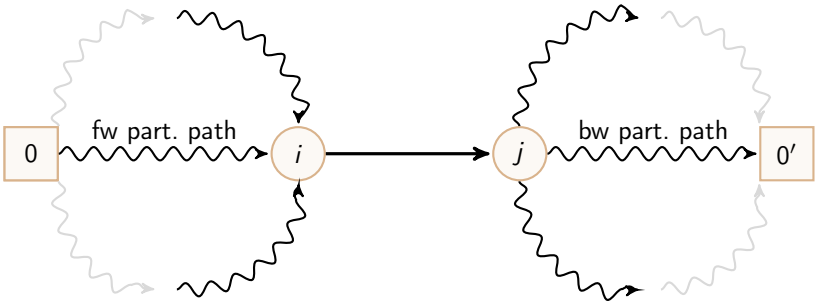


Reduced Cost-based Arc Fixing

Consider an arc $(i, j) \in A$ and the set $P[ij]$ of all paths that contain the arc (i, j).



Consider an arc $(i, j) \in A$ and the set $P[ij]$ of all paths that contain the arc (i, j).



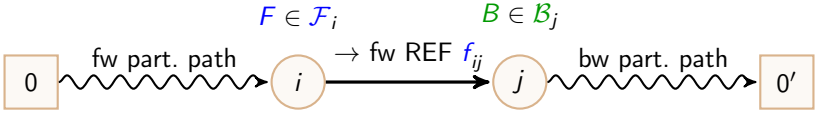
Proposition 2 (Irnich *et al.* (2010))

If the reduced cost $\tilde{c}_p(\pi)$ of all variables λ_p for $p \in P[ij]$ fulfill $\tilde{c}_p(\pi) \geq UB - LB(\pi)$, then the arc (i, j) is not used in an optimal solution and can be eliminated.

The values $\tilde{c}[ij](\pi)$ can be effectively computed for all arcs $(i,j) \in A$ with the help of:

- Full forward and full backward labeling gives label sets $(\mathcal{F}_i)_{i \in V}$ and $(\mathcal{B}_i)_{i \in V}$

$$\tilde{c}[ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B)$$

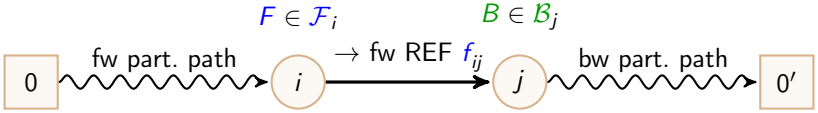


Reduced Cost-based Variable Elimination/Fixing

The values $\tilde{c}[ij](\pi)$ can be effectively computed for all arcs $(i,j) \in A$ with the help of:

- Full forward and full backward labeling gives label sets $(\mathcal{F}_i)_{i \in V}$ and $(\mathcal{B}_i)_{i \in V}$
- The forward REFs f_{ij}

$$\tilde{c}[ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B)$$

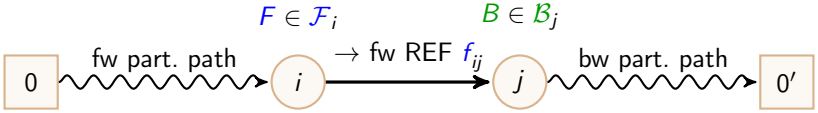


Reduced Cost-based Variable Elimination/Fixing

The values $\tilde{c}[ij](\pi)$ can be effectively computed for all arcs $(i,j) \in A$ with the help of:

- Full forward and full backward labeling gives label sets $(\mathcal{F}_i)_{i \in V}$ and $(\mathcal{B}_i)_{i \in V}$
- The forward REFs f_{ij}
- The merge operator m (return value of m is the reduced cost);

$$\tilde{c}[ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B)$$

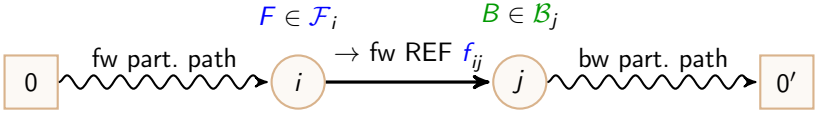


Reduced Cost-based Variable Elimination/Fixing

The values $\tilde{c}[ij](\pi)$ can be effectively computed for all arcs $(i,j) \in A$ with the help of:

- Full forward and full backward labeling gives label sets $(\mathcal{F}_i)_{i \in V}$ and $(\mathcal{B}_i)_{i \in V}$
- The forward REFs f_{ij}
- The merge operator m (return value of m is the reduced cost);
- We set $f_{ij}(F) = -b_{ij}(B) = \infty$, if infeasible; $m(\cdot,\cdot) = \infty$, if infeasible

$$\tilde{c}[ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B)$$

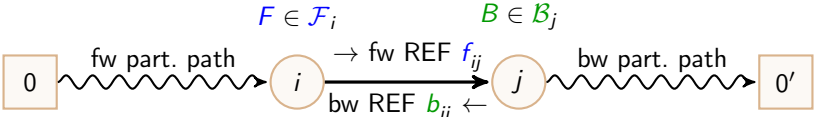


Reduced Cost-based Variable Elimination/Fixing

The values $\tilde{c}[ij](\pi)$ can be effectively computed for all arcs $(i,j) \in A$ with the help of:

- Full forward and full backward labeling gives label sets $(\mathcal{F}_i)_{i \in V}$ and $(\mathcal{B}_i)_{i \in V}$
- The forward REFs f_{ij} (the backward REFs b_{ij});
- The merge operator m (return value of m is the reduced cost);
- We set $f_{ij}(F) = -b_{ij}(B) = \infty$, if infeasible; $m(\cdot,\cdot) = \infty$, if infeasible

$$\tilde{c}[ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j}} m(F, b_{ij}(B))$$



Reduced Cost-based Variable Elimination/Fixing

Results for VRPTW (Desaulniers *et al.*, 2020), Solomon instances with 25, 50, and 100 customers. Only instances not solved in the root node; also grouped by simple (< 60 sec.) and more difficult (≥ 60 sec.).

Reduced Cost-based Variable Elimination/Fixing

Results for VRPTW (Desaulniers *et al.*, 2020), Solomon instances with 25, 50, and 100 customers. Only instances not solved in the root node; also grouped by simple (< 60 sec.) and more difficult (≥ 60 sec.).

At the root node, the fixing is performed repeatedly before/after adding valid inequalities.

Reduced Cost-based Variable Elimination/Fixing

Results for VRPTW (Desaulniers *et al.*, 2020), Solomon instances with 25, 50, and 100 customers. Only instances not solved in the root node; also grouped by simple (< 60 sec.) and more difficult (≥ 60 sec.).

At the root node, the fixing is performed repeatedly before/after adding valid inequalities.

		SAF Number of arcs		
			Fixed	
Group	#Inst	$\|A\|$	LP, %	LP+Cut, %
VRPTW				
25	16	531	75.5	82.8
50	28	1,924	61.8	86.1
100	36	7,580	71.0	93.2
< 60s	57	2,801	69.1	87.8
≥ 60s	23	7,634	66.9	90.5
All	80	4,190	68.4	88.5

More general: Paths $P[prop]$ is the subset of all P that fulfill a given property *prop*.

$$\tilde{c}[prop](\pi) = \min_{p \in P[prop]} \tilde{c}_p(\pi)$$

More general: Paths $P[prop]$ is the subset of all P that fulfill a given property $prop$.

$$\tilde{c}[prop](\pi) = \min_{p \in P[prop]} \tilde{c}_p(\pi)$$

A second property:

$[hij]$: For two arcs $(h,i),(i,j) \in A$, the path includes the sequence (h,i,j) at least once

More general: Paths $P[prop]$ is the subset of all P that fulfill a given property $prop$.

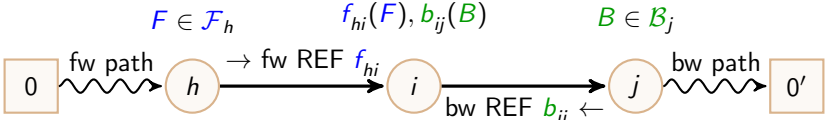
$$\tilde{c}[prop](\pi) = \min_{p \in P[prop]} \tilde{c}_p(\pi)$$

A second property:

$[hij]$: For two arcs $(h,i),(i,j) \in A$, the path includes the sequence (h,i,j) at least once

Two-arc fixing was suggested by Desaulniers *et al.* (2020):

$$\tilde{c}[hij](\pi) = \min_{\substack{F \in \mathcal{F}_h, \\ B \in \mathcal{B}_j}} m(f_{hi}(F), b_{ij}(B))$$



More general: Paths $P[prop]$ is the subset of all P that fulfill a given property $prop$.

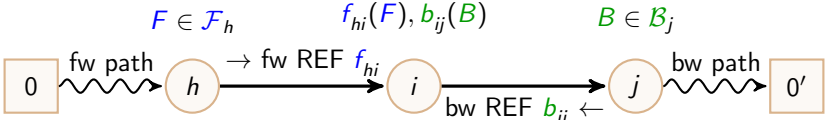
$$\tilde{c}[prop](\pi) = \min_{p \in P[prop]} \tilde{c}_p(\pi)$$

A second property:

$[hij]$: For two arcs $(h,i),(i,j) \in A$, the path includes the sequence (h,i,j) at least once

Two-arc fixing was suggested by Desaulniers *et al.* (2020):

$$\tilde{c}[hij](\pi) = \min_{\substack{F \in \mathcal{F}_h, \\ B \in \mathcal{B}_j}} m(f_{hi}(F), b_{ij}(B))$$



- Two-arc sequences cannot be eliminated from the network
- Must be eliminated during label extension
- Modified dominance comparison between labels required

Reduced Cost-based Two-Arc Fixing

Results for VRPTW (Desaulniers *et al.*, 2020), Solomon instances with 25, 50, and 100 customers. Only instances not solved in the root node; also grouped by simple (< 60 sec.) and more difficult (≥ 60 sec.).

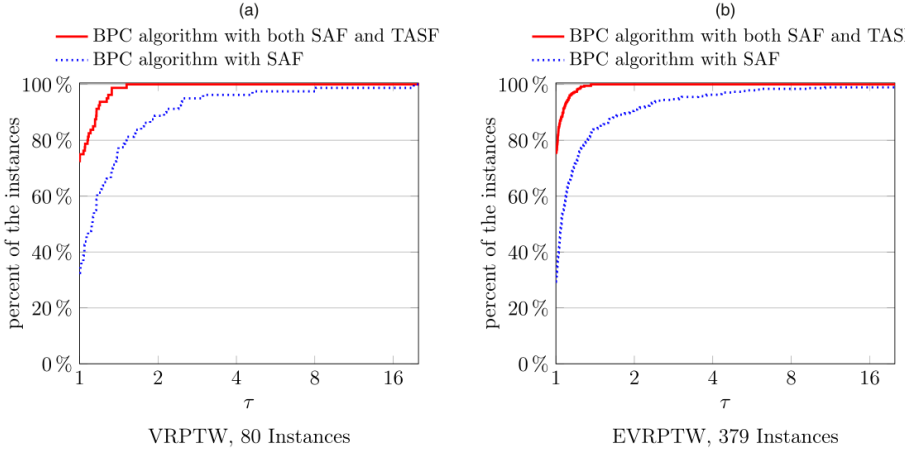
^2A two arc sequences (h, i, j) after preprocessing

$^2\hat{A}$ two arc sequences (h, i, j) after preprocessing + SAF

		SAF Number of arcs			TASF Number of two-arc sequences				
			Fixed			After SAF LP		After LP+Cut	
Group	#Inst	$\lvert A\rvert$	LP, %	LP+Cut, %	$\lvert {}^2A\rvert$	$\lvert {}^2\hat{A}\rvert$	Fixed, %	$\lvert {}^2\hat{A}\rvert$	Fixed, %
VRPTW									
25	16	531	75.5	82.8	9,075	456	37.8	220	38.4
50	28	1,924	61.8	86.1	60,429	6,201	39.9	1,167	43.5
100	36	7,580	71.0	93.2	476,558	46,103	56.0	2,475	50.1
< 60s	57	2,801	69.1	87.8	111,152	8,349	41.0	752	40.3
≥ 60s	23	7,634	66.9	90.5	550,332	59,337	61.1	3,583	60.1
All	80	4,190	68.4	88.5	237,416	23,008	46.0	1,566	45.2

Similar results for EVRPTW.

Figure 1. (Color online) Performance Profiles $\rho_A(\tau)$ for Algorithms $A \in \mathcal{A}$ = {BPC with SAF, BPC with SAF and TASF}



New properties:

$[R_i^{time} < t, ij]$: All paths that include arc (i, j) and allow a start of service at vertex i before time t followed by the traversal of arc (i, j);

New properties:

$[R_i^{time} < t, ij]$: All paths that include arc (i, j) and allow a start of service at vertex i before time t followed by the traversal of arc (i, j);

$[ij, R_j^{time} < t]$: All paths that include arc (i, j) and allow a start of service at vertex j before time t after the traversal of arc (i, j).

Likewise for $> t$.

New properties:

$[R_i^{time} < t, ij]$: All paths that include arc (i,j) and allow a start of service at vertex i before time t followed by the traversal of arc (i,j);

$[ij, R_j^{time} < t]$: All paths that include arc (i,j) and allow a start of service at vertex j before time t after the traversal of arc (i,j).

Likewise for $> t$.

$$\tilde{c}[R_i^{time} < t, ij](\pi) = \min_{\substack{F \in \mathcal{F}_i : F^{time} < t \\ B \in \mathcal{B}_j}} m(F, b_{ij}(B))$$

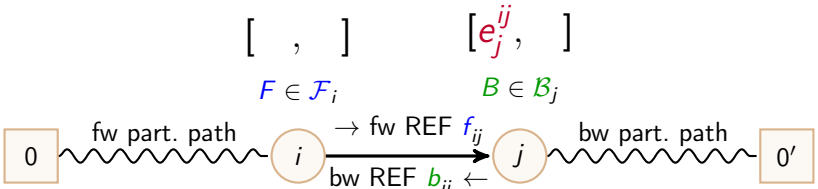
$$\tilde{c}[R_i^{time} > t, ij](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j : b_{ij}^{time}(B) > t}} m(F, b_{ij}(B))$$

$$\tilde{c}[ij, R_j^{time} < t](\pi) = \min_{\substack{F \in \mathcal{F}_i : f_{ij}^{time}(F) < t \\ B \in \mathcal{B}_j}} m(f_{ij}(F), B)$$

$$\tilde{c}[ij, R_j^{time} > t](\pi) = \min_{\substack{F \in \mathcal{F}_i, \\ B \in \mathcal{B}_j : B^{time} > t}} m(f_{ij}(F), B)$$

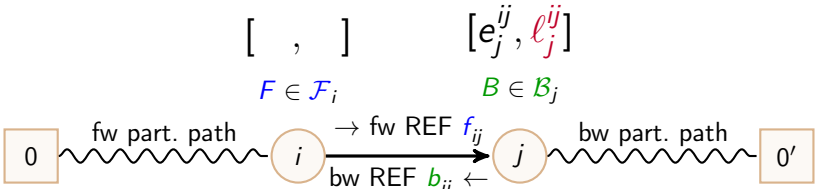
Redefine the time-related parts of forward and backward REFs with the four arc-specific attributes $e_i^{ij}, e_j^{ij}, \ell_i^{ij}$, and ℓ_j^{ij} in the following way:

$$F_j^{time} = f_{ij}^{time}(F_i) = \max\{e_j^{ij}, F_i^{time} + t_{ij}\}$$



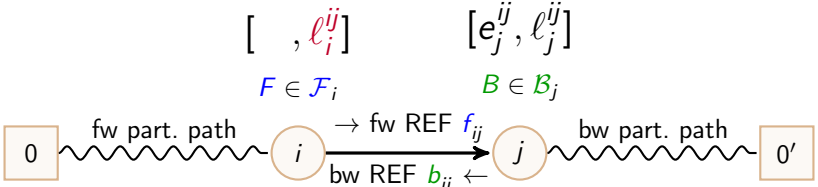
Redefine the time-related parts of forward and backward REFs with the four arc-specific attributes $e_i^{ij}, e_j^{ij}, \ell_i^{ij}$, and ℓ_j^{ij} in the following way:

$$F_j^{time} = f_{ij}^{time}(F_i) = \max\{e_j^{ij}, F_i^{time} + t_{ij}\} \qquad \text{feasible if} \quad F_j^{time} \leq \ell_j^{ij}$$



Redefine the time-related parts of forward and backward REFs with the four arc-specific attributes $e_i^{ij}, e_j^{ij}, \ell_i^{ij}$, and ℓ_j^{ij} in the following way:

$$F_j^{time} = f_{ij}^{time}(F_i) = \max\{e_j^{ij}, F_i^{time} + t_{ij}\} \qquad \text{feasible if} \quad F_j^{time} \leq \ell_j^{ij}$$
$$B_i^{time} = b_{ij}^{time}(B_j) = \min\{\ell_i^{ij}, B_j^{time} - t_{ij}\}$$



Redefine the time-related parts of forward and backward REFs with the four arc-specific attributes $e_i^{ij}, e_j^{ij}, \ell_i^{ij}$, and ℓ_j^{ij} in the following way:

$$F_j^{time} = f_{ij}^{time}(F_i) = \max\{e_j^{ij}, F_i^{time} + t_{ij}\} \qquad \text{feasible if} \quad F_j^{time} \leq \ell_j^{ij}$$

$$B_i^{time} = b_{ij}^{time}(B_j) = \min\{\ell_i^{ij}, B_j^{time} - t_{ij}\} \qquad \text{feasible if} \quad B_i^{time} \geq e_i^{ij}$$

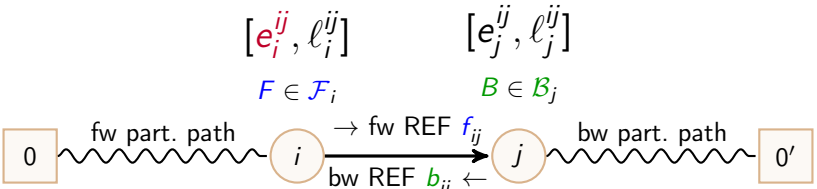
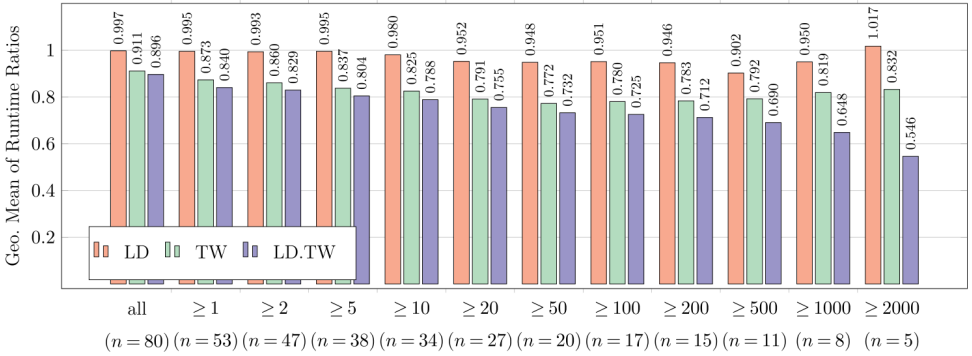


Figure 2. (Color online) Geometric Mean of the Computation Time Ratios w.r.t. to the Baseline Setting AF (Arc Fixing) for the Resource Window Reduction Settings LD, TW, and LD.TW for the VRPTW; Results Are Grouped According to Running Times, Where $\geq rt$ Indicates that Only Instances with a Computation Time t_{AF} of at least rt Are Considered

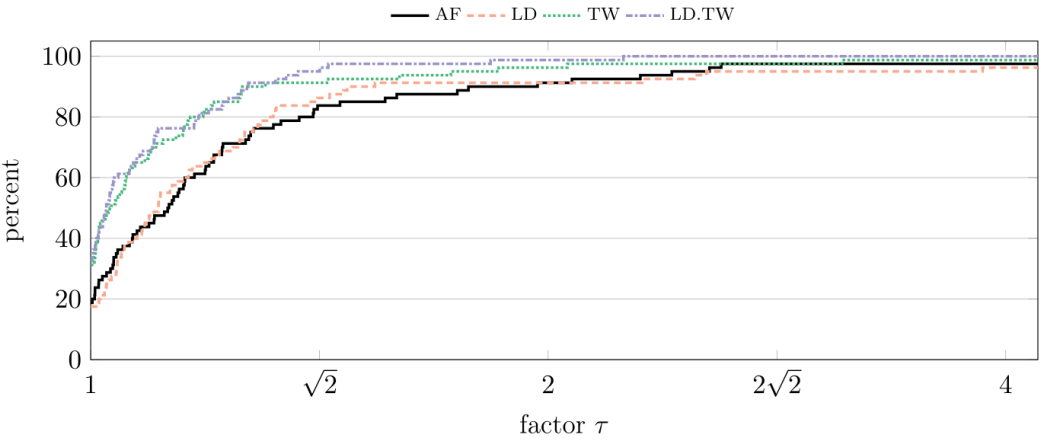


LD load resource with resource window $[0, Q]$

TW time resource with resource window $[e_i, \ell_i]$

LD.TW both

Arc-Specific Resource Windows Reduction



BPC algorithm using:

- AF (single) arc fixing only
- LD AF + load resource with resource window $[0, Q]$
- TW AF + time resource with resource window $[e_i, \ell_i]$
- LD.TW AF and both

Conclusions

Many more applications for BPC:

- Parallel machine scheduling (van den Akker *et al.*, 1999)
- Multi-activity shift scheduling (Boyer *et al.*, 2013)
- Graph problems
  - Vertex coloring (Held *et al.*, 2012)
  - Community detection (Aloise *et al.*, 2010a)
  - ...
- Clustering (Aloise *et al.*, 2010b)
- ...
- (many more)

BPC Algorithms:

- Powerful algorithmic technique to solve practically relevant problems

BPC Algorithms:

- Powerful algorithmic technique to solve practically relevant problems
- Many different algorithmic components in an advanced BPC algorithm

BPC Algorithms:

- Powerful algorithmic technique to solve practically relevant problems
- Many different algorithmic components in an advanced BPC algorithm
- BPC algorithms can ...
  1. deliver strong bounds,
  2. eliminate inherent symmetry ($\rightarrow$ Aggregation)
  3. cope with non-linearities ($\rightarrow$ using DP, SAT, ...), and
  4. benefit from effective algorithms for subproblems ($\rightarrow$ SPPRC, Benders, ...).
  5. Software frameworks available ($\rightarrow$ SCIP, VRPSolver)

BPC Algorithms:

- Powerful algorithmic technique to solve practically relevant problems
- Many different algorithmic components in an advanced BPC algorithm
- BPC algorithms can ...
  1. deliver strong bounds,
  2. eliminate inherent symmetry ($\rightarrow$ Aggregation)
  3. cope with non-linearities ($\rightarrow$ using DP, SAT, ...), and
  4. benefit from effective algorithms for subproblems ($\rightarrow$ SPPRC, Benders, ...).
  5. Software frameworks available ($\rightarrow$ SCIP, VRPSolver)

The road ahead is probably the interaction with ML methods...

- Automated BPC algorithm configuration
- Automatic Dantzig-Wolfe reformulation (`GCG`)
- Solving ML problems with BPC algorithms

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., and Liberti, L. (2010a). Column generation algorithms for exact modularity maximization in networks. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, **82**(4 Pt 2), 046112.

Aloise, D., Hansen, P., and Liberti, L. (2010b). An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, **131**(1–2), 195–220.

Altman, C., Desaulniers, G., and Errico, F. (2023). The fragility-constrained vehicle routing problem with time windows. *Transportation Science*, **57**(2), 552–572.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, **115**(2), 351–385.

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.

Bianchessi, N., Irnich, S., and Tilk, C. (2021). A branch-price-and-cut algorithm for the capacitated multiple vehicle traveling purchaser problem with unitary demand. *Discrete Applied Mathematics*, **288**, 152–170.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, **49**(2), 369–383.

Boland, N., Dethridge, J., and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, **34**(1), 58 – 68.

Boyer, V., Gendron, B., and Rousseau, L.-M. (2013). A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. *Journal of Scheduling*, **17**(2), 185–197.

Bulhões, T., Sadykov, R., and Uchoa, E. (2018). A branch-and-price algorithm for the minimum latency problem. *Computers & Operations Research*, **93**, 66–78.

Cherkesly, M., Desaulniers, G., and Laporte, G. (2015). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, **49**(4), 752–766.

Christofides, N., Mingozzi, A., and Toth, P. (1981). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, **11**(2), 145–164.

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.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, **26**(1), 88–102.

Contardo, C., Desaulniers, G., and Lessard, F. (2015). Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks*, **65**(1), 88–99.

Dabia, S., Ropke, S., van Woensel, T., and Kok, T. D. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, **47**(3), 380–396.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, **58**(1), 179–192.

Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, **42**(3), 387–404.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, **64**(6), 1388–1405.

Desaulniers, G., Pecin, D., and Contardo, C. (2017). Selective pricing in branch-price-and-cut algorithms for vehicle routing. *EURO Journal on Transportation and Logistics*.

Desaulniers, G., Gschwind, T., and Irnich, S. (2020). Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science*, **54**(5), 1170–1188.

Desrosiers, J., Lübbecke, M., Desaulniers, G., and Gauthier, J. B. (2024). *Branch-and-Price*. Unpublished.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, **42**(5), 977–978.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR: Information Systems and Operational Research*, **45**(4), 239–256.

Florio, A. M., Hartl, R. F., and Minner, S. (2020). New exact algorithm for the vehicle routing problem with stochastic demands. *Transportation Science*, **54**(4), 1073–1090.

Florio, A. M., Feillet, D., Poggi, M., and Vidal, T. (2022). Vehicle routing with stochastic demands and partial reoptimization. *Transportation Science*, **56**(5), 1393–1408.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, **47**(2), 247–263.

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**(6), 849–859.

Goel, A. and Irnich, S. (2016). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, **51**(2), 737–754.

Gschwind, T. (2015). A comparison of column-generation approaches to the synchronized pickup and delivery problem. *European Journal of Operational Research*, **247**(1), 60–71.

Gschwind, T. and Irnich, S. (2015). Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, **49**(2), 335–354.

Halse, K. (1992). *Modelling and solving complex vehicle routing problems*. Ph.d. dissertation, IMSOR, Technical University of Denmark, Lyngby, Denmark.

Held, S., Cook, W., and Sewell, E. C. (2012). Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, **4**(4), 363–381.

Himmich, I., El Hallaoui, I., and Soumis, F. (2024). A multiphase dynamic programming algorithm for the shortest path problem with resource constraints. *European Journal of Operational Research*, **315**(2), 470–483.

Houck, D., Picard, J., Queyranne, M., and Vemuganti, R. (1980). The travelling salesman problem as a constrained shortest path problem: theory and computational experience. *Opsearch*, **17**, 93–109.

Ioachim, I., Gélinas, S., Desrosiers, J., and Soumis, F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, **31**, 193–204.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

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.

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.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Lam, E., Desaulniers, G., and Stuckey, P. J. (2022). Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. *Computers and Operations Research*, **145**, 105870.

Montoya, A., Guéret, C., Mendoza, J. E., and Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, **103**, 87–110.

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., and Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, **53**(4), 1043–1066.

Nemhauser, G. and Wolsey, L. (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc.

Parragh, S. N., Cordeau, J.-F., Doerner, K. F., and Hartl, R. F. (2012). Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum*, **34**(3), 593–633.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., and Santos, H. (2017a). Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, **45**(3), 206–209.

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

Prescott-Gagnon, E., Desaulniers, G., Drexl, M., and Rousseau, L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science*, **44**(4), 455–473.

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.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, **51**(3), 155–170.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, **43**(3), 267–286.

Rostami, B., Desaulniers, G., Errico, F., and Lodi, A. (2021). Branch-price-and-cut algorithms for the vehicle routing problem with stochastic and correlated travel times. *Operations Research*, **69**(2), 436–455.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2017). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*. Forthcoming.

Sadykov, R. (2019). *Modern Branch-Cut-and-Price*. Habilitation thesis, Institut de Mathématiques de Bordeaux, Université de Bordeaux, Bordeaux.

Sadykov, R., Uchoa, E., and Pessoa, A. (2021). A bucket graph–based labeling algorithm with application to vehicle routing. *Transportation Science*, **55**(1), 4–28.

Scheithauer, G. (2018). *Introduction to Cutting and Packing Optimization*. Springer International Publishing.

Scheithauer, G. and Terno, J. (1992). *About the gap between the optimal values of the integer and continuous relaxation of the one-dimensional cutting stock problem*, pages 439–444. Springer, Berlin/Heidelberg.

Spliet, R. (2023). Technical note—the complexity of the pricing problem of the set partitioning formulation of vehicle routing problems. *Operations Research*, **71**(5), 1454–1457.

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.

van den Akker, J. M., Hoogeveen, J. A., and van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research*, **47**(6), 862–872.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, **48**(1), 111–128.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, **165**(1), 97–107.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2020). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*, **32**(2), 428–443.