Component Bound Branching in a Branch-and-Price Framework

Til Mohr

til.mohr@rwth-aachen.de

Master thesis at the
Chair of Operations Research @ RWTH Aachen
www.or.rwth-aachen.de

September 5, 2024

Table of Contents

▶ consider the following bounded integer program (*IP*):

$$
\begin{aligned}
z_{IP}^* = \min \quad & \boldsymbol{c}^\mathsf{T} \boldsymbol{x} \\
\text{s.t.} \quad & \boldsymbol{Ax} \geq \boldsymbol{b} && [\boldsymbol{\pi_b}] \\
& \boldsymbol{Dx} \geq \boldsymbol{d} && [\boldsymbol{\pi_d}] \\
& \boldsymbol{x} \in \mathbb{Z}_+^n
\end{aligned}
$$

▶ we want to solve this *IP* using Column Generation (CG)

▶ consider the following bounded integer program (*IP*):

$$
\begin{aligned}
z_{IP}^* = \min \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\
\text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \quad [\boldsymbol{\pi_b}] \\
& \boldsymbol{D}\boldsymbol{x} \geq \boldsymbol{d} \quad [\boldsymbol{\pi_d}] \\
& \boldsymbol{x} \in \mathbb{Z}_+^n
\end{aligned}
$$

▶ we want to solve this *IP* using Column Generation (CG)

$\rightarrow$ Apply Dantzig-Wolfe Reformulation

Dantzig-Wolfe Reformulation for *IP*s

▶ reformulation using discretization yields the master problem (*MP*):

$$z_{MP}^* = \min \quad \sum_{q \in Q} c_q \lambda_q$$

$$\text{s.t.} \quad \sum_{q \in Q} \boldsymbol{a}_q \lambda_q \geq \boldsymbol{b} \qquad [\boldsymbol{\pi_b}]$$

$$\sum_{q \in Q} \lambda_q = 1 \qquad [\pi_0]$$

$$\lambda_q \in \{0, 1\} \qquad \forall q \in Q$$

▶ and the subproblem (*SP*):

$$z_{SP}^* = \min \quad \left(\boldsymbol{c}^\intercal - \boldsymbol{\pi_b}^\intercal \boldsymbol{A} \right) \boldsymbol{x} - \pi_0$$

$$\text{s.t.} \quad \boldsymbol{D} \boldsymbol{x} \geq \boldsymbol{d} \qquad [\boldsymbol{\pi_d}]$$

$$\boldsymbol{x} \in \mathbb{Z}_+^n$$

Branch-and-Price

Branch-and-Price Algorithm:

1. solve the relaxed *MP* using CG to optimality
2. branch, if $\boldsymbol{\lambda}^*$ is fractional
3. repeat

Branch-and-Price

Branch-and-Price Algorithm:

1. solve the relaxed *MP* using CG to optimality
2. branch, if $\boldsymbol{\lambda}^*$ is fractional
3. repeat

But how to branch?

▶ branching on a single λ_q is unbalanced:
   $\lambda_q \leq 0$ forbids almost nothing, $\lambda_q \geq 1$ forbids much

Branch-and-Price

Branch-and-Price Algorithm:

1. solve the relaxed *MP* using CG to optimality
2. branch, if $\boldsymbol{\lambda}^*$ is fractional
3. repeat

But how to branch?

▶ branching on a single λ_q is unbalanced:
  $\lambda_q \leq 0$ forbids almost nothing, $\lambda_q \geq 1$ forbids much

→ branch on a set of columns $Q' \subset Q$, such that:

$$\sum_{q \in Q'} \lambda_q^* =: K \notin \mathbb{Z}$$

Component Bound Sequences

▶ to find such a set Q' of columns, we find a component bound sequence:

$$S := \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

Component Bound Sequences

- to find such a set Q' of columns, we find a component bound sequence:

$$S \coloneqq \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

- a master variable λ_q satisfies a component bound (x_i, η_i, v_i) if:

$$x_{q,i} \; \eta_i \; v_i$$

- we define Q' as the set of columns, that satisfy all component bounds in S
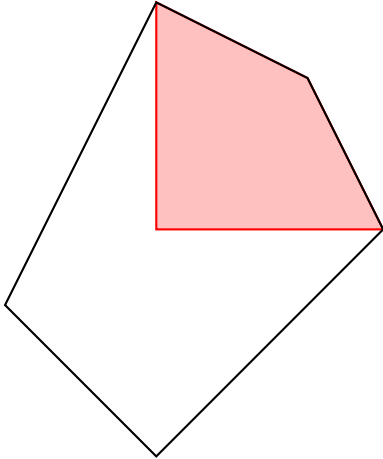- there always exists a component bound sequence S such that:

$$\sum_{q \in Q'} \lambda_q^* = K \notin \mathbb{Z}$$

Component Bound Sequences

- to find such a set Q' of columns, we find a component bound sequence:

$$S := \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

- a master variable λ_q satisfies a component bound (x_i, η_i, v_i) if:

$$x_{q,i} \ \eta_i \ v_i$$

- we define Q' as the set of columns, that satisfy all component bounds in S
- there always exists a component bound sequence S such that:

$$\sum_{q \in Q'} \lambda_q^* = K \notin \mathbb{Z}$$

- how do we enforce this?

Table of Contents

- assume we have found a component bound Sequence
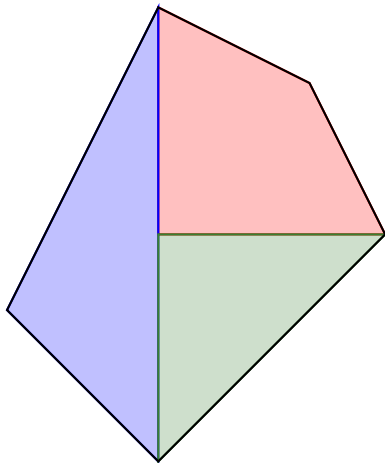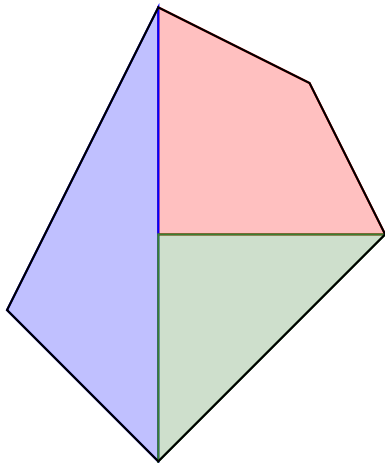 $S = \{(x_1, \geq, 1), (x_2, \geq, 2)\}$

Overview of Vanderbeck's Generic Branching Scheme

- ▶ assume we have found a component bound Sequence
  $S = \{(x_1, \geq, 1), (x_2, \geq, 2)\}$

- ▶ create $|S| + 1 = 3$ child nodes:
  1. $S_1 := \{(x_1, <, 1)\}$
  2. $S_2 := \{(x_1, \geq, 1), (x_2, <, 2)\}$
  3. $S_3 := \{(x_1, \geq, 1), (x_2, \geq, 2)\}$

- ▶ search for solutions only within these regions

- assume we have found a component bound Sequence
  $S = \{(x_1, \geq, 1), (x_2, \geq, 2)\}$

- create $|S| + 1 = 3$ child nodes:
  1. $S_1 := \{(x_1, <, 1)\}$
  2. $S_2 := \{(x_1, \geq, 1), (x_2, <, 2)\}$
  3. $S_3 := \{(x_1, \geq, 1), (x_2, \geq, 2)\}$

- search for solutions only within these regions

- refine the regions in deeper nodes

▶ to the *MP* add constraint:

$$\sum_{q \in Q'_j} \lambda_q \geq K_j$$

▶ move the bounds in the *SP*:

$$x_i \geq v_i \quad \forall\,(x_i, \geq, v_i) \in S_j$$
$$x_i \leq v_i \quad \forall\,(x_i, \leq, v_i) \in S_j$$

Table of Contents

Component Bound Branching Rule

Again:

- find a subset Q' of columns using a component bound sequence S:

$$S := \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

- such that: $\quad \sum_{q \in Q'} \lambda_q^* = K \notin \mathbb{Z}$

Component Bound Branching Rule

Again:

- find a subset Q' of columns using a component bound sequence S:

$$S := \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

- such that: $\quad \sum_{q \in Q'} \lambda_q^* = K \notin \mathbb{Z}$

Alternative scheme:

- create binary search tree:

$$\sum_{q \in Q'} \lambda_q \leq \lfloor K \rfloor \quad [\gamma] \qquad\qquad \sum_{q \in Q'} \lambda_q \geq \lceil K \rceil \quad [\gamma] \qquad (1)$$

Component Bound Branching Rule

Again:

▶ find a subset Q' of columns using a component bound sequence S:

$$S := \{(x_i, \eta_i, v_i) \mid \eta_i \in \{\leq, \geq\}, v_i \in \mathbb{Z}\}$$

▶ such that: $\quad \sum_{q \in Q'} \lambda_q^* = K \notin \mathbb{Z}$

Alternative scheme:

▶ create binary search tree:

$$\sum_{q \in Q'} \lambda_q \leq \lfloor K \rfloor \quad [\gamma] \qquad\qquad \sum_{q \in Q'} \lambda_q \geq \lceil K \rceil \quad [\gamma] \qquad (1)$$

▶ allows for solutions violating S to be found / generated

Component Bound Branching Rule

- SP determines whether a new column is in Q', i.e., satisfies S
- create new variable $y \in \{0, 1\}$ along new constraints, such that: $y = 1 \Leftrightarrow \boldsymbol{x} \in S$

Component Bound Branching Rule

▶ SP determines whether a new column is in Q', i.e., satisfies S
▶ create new variable $y \in \{0, 1\}$ along new constraints, such that: $y = 1 \Leftrightarrow \boldsymbol{x} \in S$

$$
\begin{aligned}
z_{SP}^* = \min \quad & \left(\boldsymbol{c}^\mathsf{T} - \boldsymbol{\pi}_b^\mathsf{T} \boldsymbol{A}\right) \boldsymbol{x} - \gamma y - \pi_0 \\
\text{s.t.} \quad & \boldsymbol{D}\boldsymbol{x} \geq \boldsymbol{d} \\
& y = 1 \Leftrightarrow \sum_{i \in S} y_i = |S| \\
& y_i = 1 \Leftrightarrow x_i \leq v_i && \forall\, (x_i, \eta_i, v_i) \in S \\
& y \in \{0, 1\} \\
& y_i \in \{0, 1\} && \forall\, (x_i, \eta_i, v_i) \in S \\
& \boldsymbol{x} \in \mathbb{Z}_+^n
\end{aligned}
$$

Table of Contents

Master Constraints without Original Formulation

▶ `GCG` solver only allows adding original constraints, which are reformulated
▶ we want to add constraint *MP*:

$$\sum_{p \in P} \mathrm{f}(p)\lambda_p + \sum_{r \in R} \mathrm{f}(r)\lambda_r \leq f \qquad [\gamma]$$

▶ with the following modification to the pricing problem:

$$
\begin{aligned}
z_{SP}^* = \min \quad & \left(\boldsymbol{c}^\intercal - \boldsymbol{\pi}_b^\intercal \boldsymbol{A}\right)\boldsymbol{x} - \gamma y - \pi_0 \\
\text{s.\,t.} \quad & \boldsymbol{D}\boldsymbol{x} \geq \boldsymbol{d} \qquad [\boldsymbol{\pi_d}] \\
& y = \mathrm{f}(\boldsymbol{x}) \\
& \boldsymbol{x} \in \mathbb{Z}_+^n \\
& y \in Y
\end{aligned}
$$

Master Constraints without Original Formulation

▶ `GCG` solver only allows adding original constraints, which are reformulated

▶ we want to add constraint *MP*:

$$\sum_{p \in P} \mathrm{f}(p)\lambda_p + \sum_{r \in R} \mathrm{f}(r)\lambda_r \leq f \qquad [\gamma]$$

▶ with the following modification to the pricing problem:

$$
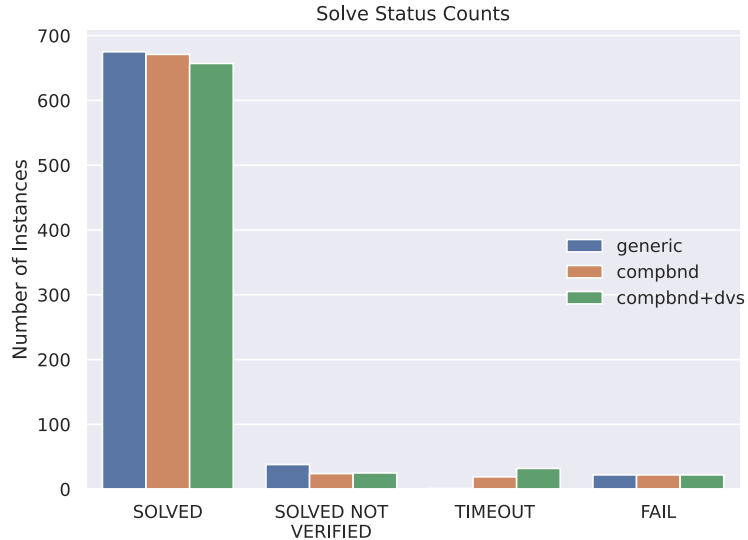\begin{aligned}
z_{SP}^* = \min \quad & \left(\boldsymbol{c}^\intercal - \boldsymbol{\pi}_b^\intercal \boldsymbol{A}\right)\boldsymbol{x} - \gamma y - \pi_0 \\
\text{s.t.} \quad & \boldsymbol{D}\boldsymbol{x} \geq \boldsymbol{d} \qquad [\boldsymbol{\pi_d}] \\
& y = \mathrm{f}(\boldsymbol{x}) \\
& \boldsymbol{x} \in \mathbb{Z}_+^n \\
& y \in Y
\end{aligned}
$$

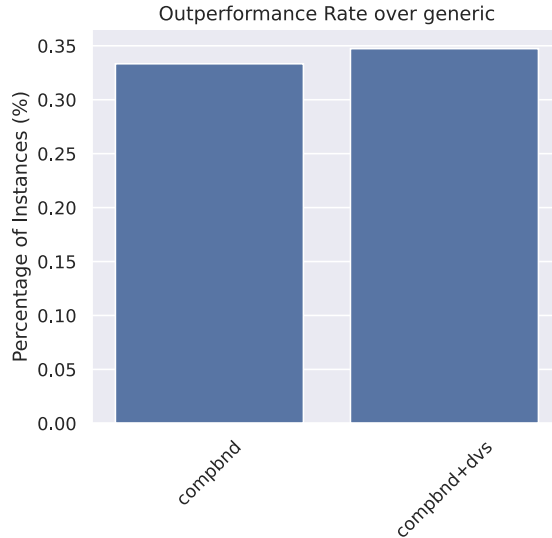$\rightarrow$ new interface in `GCG` to create and manage such constraints

Table of Contents
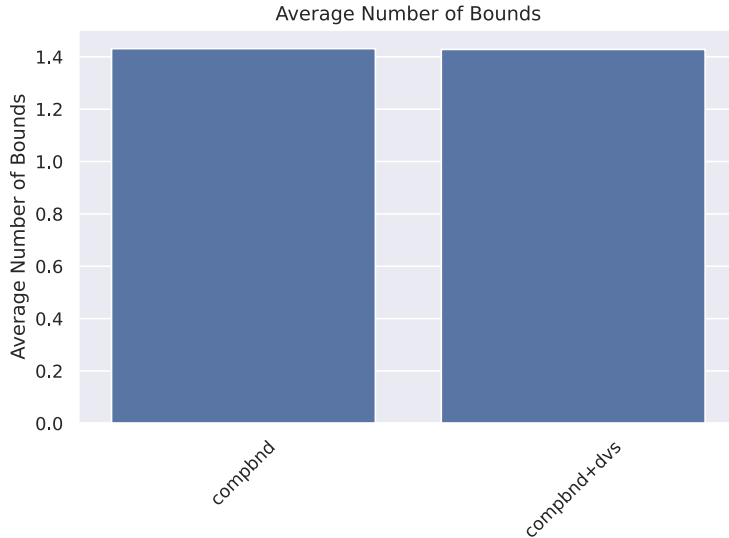
Core Evaluation Questions

- ▶ Vanderbeck's Generic Branching vs. Component Bound Branching Rule
- ▶ Dual Value Stabilization: Root-Only vs. Full-Tree
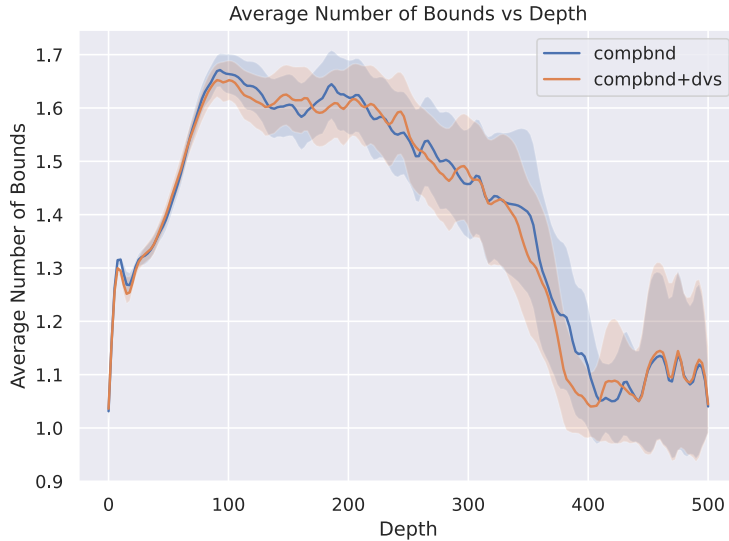- ▶ How many Component Bounds are required?

Solve Status Counts

Evaluation

Solving Time (s)

Outperformance Rate over generic

Evaluation

Average Number of Bounds

Evaluation

Average Number of Bounds vs Depth

Table of Contents

Conclusion

- implemented a new interface for managing generic mastercuts in `GCG`
- implemented a simple branching rule for component bound sequences using generic mastercuts
- Vanderbeck's Generic Branching Scheme outperforms the Component Bound Branching Rule
- implemenation of the Component Bound Branching Rule showcases the potential of the new interface
- $\rightarrow$ opens up new possibilities for future research (e.g., master separators)

References I

Achterberg, Tobias (2007). "Constraint integer programming".

Achterberg, Tobias, Thorsten Koch, and Alexander Martin (2005). "Branching rules revisited". *Operations Research Letters* 33.1, pp. 42–54.

Bastubbe, Michael, Alexander Helber, *et al.* (2025). "strIPlib: Structured Integer Programming Library".

Bastubbe, Michael, Marco E Lübbecke, and Jonas T Witt (2018). "A Computational Investigation on the Strength of Dantzig-Wolfe Reformulations". *17th International Symposium on Experimental Algorithms (SEA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chappell, Nathan (2019). "Minkowski-Weyl Theorem".

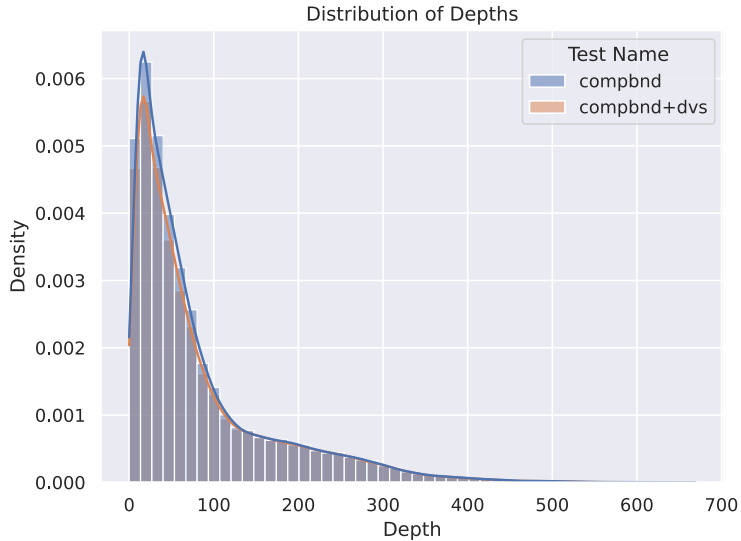Dantzig, George B and Mukund N Thapa (1997). *The simplex method*. Springer.

References II

Desrosiers, Jacques *et al.* (June 2024). *Branch-and-Price*. Les Cahiers du GERAD G-2024-36. GERAD, Montréal QC H3T 2A7, Canada: Groupe détudes et de recherche en analyse des décisions, pp. 1–689. eprint: `https://www.gerad.ca/papers/G-2024-36.pdf?locale=fr`. published.

Gamrath, Gerald (2010). "Generic branch-cut-and-price". MA thesis.

Matouek, Jií and Bernd Gärtner (2007). *Understanding and using linear programming*. Vol. 1. Springer.

Pessoa, Artur, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck (2018). "Automation and combination of linear-programming based stabilization techniques in column generation". *INFORMS Journal on Computing* 30.2, pp. 339–360.

Pessoa, Artur, Ruslan Sadykov, Eduardo Uchoa, and Francois Vanderbeck (2013). "In-out separation and column generation stabilization by dual price smoothing". *Experimental Algorithms: 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings 12*. Springer, pp. 354–365.

References III

Ploskas, Nikolaos and Nikolaos Samaras (2014). "Pivoting rules for the revised simplex algorithm". *Yugoslav Journal of Operations Research* 24.3, pp. 321–332.

Schmickerath, Marcel (2012). "Experiments on Vanderbeck's generic Branch-and-Price scheme".

Vanderbeck, François (2011). "Branching in branch-and-price: a generic scheme". *Mathematical Programming* 130, pp. 249–294.

Vanderbeck, François and Laurence A Wolsey (1996). "An exact algorithm for IP column generation". *Operations research letters* 19.4, pp. 151–159.

— (2010). *Reformulation and decomposition of integer programs*. Springer.

Witt, Jonas T (2013). "Separation of Generic Cutting Planes in Branch-and-Price". PhD thesis. Master's thesis. RWTH Aachen University.

Wolsey, Laurence A and George L Nemhauser (2014). *Integer and combinatorial optimization*. John Wiley & Sons.

Appendix

Distribution of Depths

Appendix

Distribution of Number of Bounds