Accelerating branch-and-price by heuristic pricing for integrality

Elina Rönnberg

MIP Europe 2025

LINKÖPING UNIVERSITY

What?

- Large-scale discrete optimisation: Applications where branch-and-price is a very successful method

What?

- Large-scale discrete optimisation: Applications where branch-and-price is a very successful method
- Large Neighbourhood Search (LNS): Improve computational performance of branch-and-price for difficult instances, i.e. when root-node gap is large

Why?

- LNS heuristics are vital components in generic MIP solvers
- Challenging to extend them to settings where columns are generated

Why?

- LNS heuristics are vital components in generic MIP solvers
- Challenging to extend them to settings where columns are generated
- "Standard column generation only cares about LP" $\rightarrow$ unexplored potential

How?

LNS of destroy-repair type

- Destroy method:
  Remove columns from current solution
- Repair method:
  Generate columns that benefit the integer program

How?

LNS of destroy-repair type

- Destroy method:
  Remove columns from current solution
- Repair method:
  Generate columns that benefit the integer program

Key question:
How can we price with integer solutions in mind?

Outline

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties
- Sometimes much much better properties

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties
- Sometimes much much better properties
- The number of variables increases

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties
- Sometimes much much better properties
- The number of variables increases
- Typically the number of variables explodes $\rightarrow$ solution space cannot be explicitly represented

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties
- Sometimes much much better properties

Follows from decomposition

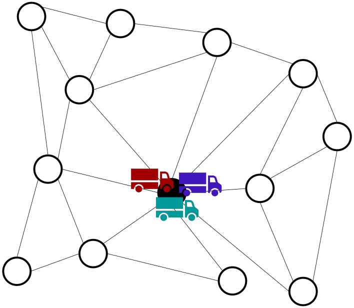
- The number of variables increases
- Typically the number of variables explodes → solution space cannot be explicitly represented

Solution method needs to handle this

Textbook example: Vehicle Routing Problems (VRP)

Problem formulation

Use these three vehicles
Visit all customers
Minimise total travel time



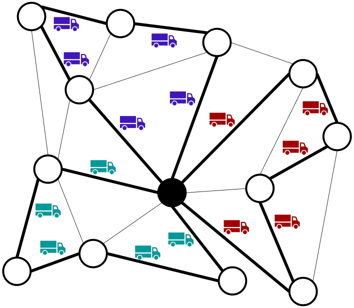
Textbook example: Vehicle Routing Problems (VRP)

Compact formulation

Decision variables:

$$x_{qk} = \begin{cases} 1 \text{ if vehicle } q \\ \quad \text{uses arc } k, \\ 0 \text{ otherwise} \end{cases}$$

Constraints:
Feasible routes for all vehicles
Vehicles cover all customers



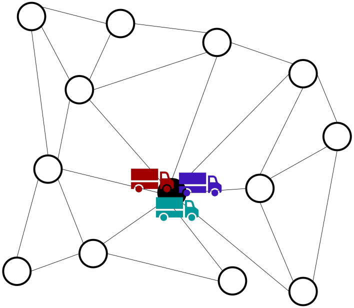
Textbook example: Vehicle Routing Problems (VRP)

Extended formulation

Enumerate **all** routes,
specify by parameter:

$$a_{ij} = \begin{cases} 1 & \text{if route } j \\ & \text{visits customer } i \\ 0 & \text{otherwise.} \end{cases}$$

Constraints:
Feasible routes for all vehicles

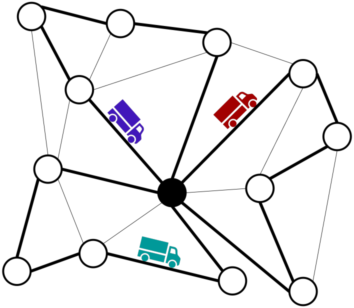


Textbook example: Vehicle Routing Problems (VRP)

Extended formulation

Decision variables:

$$\lambda_{qj} = \begin{cases} 1 \text{ if vehicle } q \\ \quad \text{uses route } j, \\ 0 \text{ otherwise.} \end{cases}$$



Constraints:
One route per vehicle
Vehicles cover all customers

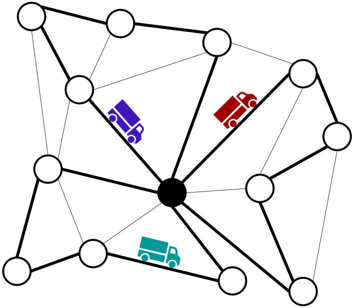
Textbook example: Vehicle Routing Problems (VRP)

Extended formulation

Decision variables:

$$\lambda_{qj} = \begin{cases} 1 \text{ if vehicle } q \\ \quad \text{uses route } j, \\ 0 \text{ otherwise.} \end{cases}$$

Constraints:
One route per vehicle
Vehicles cover all customers



**Typically not reasonable to enumerate all routes—
but for now, assume it is!**

Why make a reformulation?

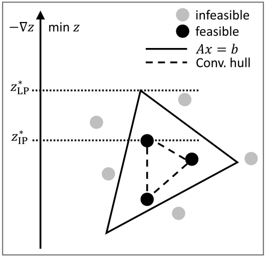
In a MIP, the strength of the formulation matters

$$z^*_{\text{IP}} = \min \; c^{\mathsf{T}}x \quad \text{s.t.} \; Ax = b, \; x \in \{0,1\}^n$$

$$z^*_{\text{LP}} = \min \; c^{\mathsf{T}}x \quad \text{s.t.} \; Ax = b, \; x \in [0,1]^n$$

Why make a reformulation?

In a MIP, the strength of the formulation matters

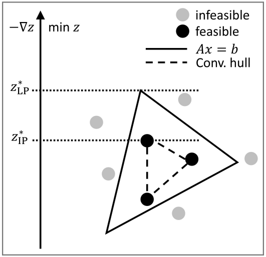


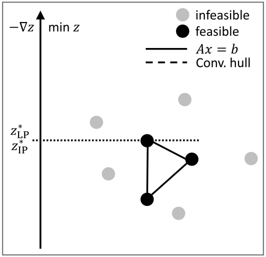
$$z^*_{\mathrm{IP}} = \min \; c^{\mathrm{T}} x \quad \text{s.t.} \quad Ax = b, \quad x \in \{0,1\}^n$$

$$z^*_{\mathrm{LP}} = \min \; c^{\mathrm{T}} x \quad \text{s.t.} \quad Ax = b, \quad x \in [0,1]^n$$

Why make a reformulation?

In a MIP, the strength of the formulation matters



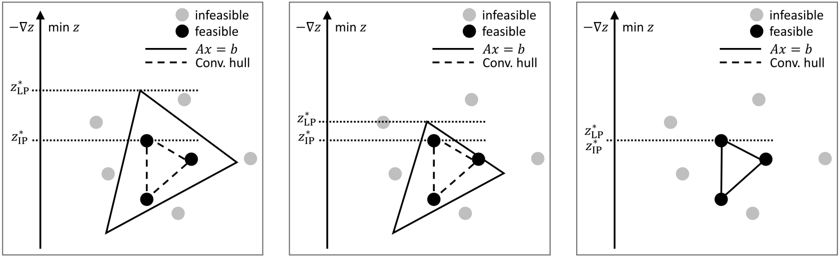


$$z^*_{\mathrm{IP}} = \min \quad c^{\mathsf{T}} x \quad \text{s.t.} \quad Ax = b, \quad x \in \{0,1\}^n$$

$$z^*_{\mathrm{LP}} = \min \quad c^{\mathsf{T}} x \quad \text{s.t.} \quad Ax = b, \quad x \in [0,1]^n$$

Why make a reformulation?

In a MIP, the strength of the formulation matters



$$z^*_{\mathrm{IP}} = \min \; c^{\mathsf{T}}x \quad \text{s.t.} \quad Ax = b, \quad x \in \{0,1\}^n$$

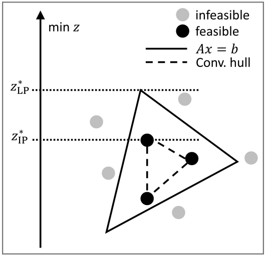
$$z^*_{\mathrm{LP}} = \min \; c^{\mathsf{T}}x \quad \text{s.t.} \quad Ax = b, \quad x \in [0,1]^n$$

Convexification

Let the LP-polytope originate from two groups of constraints $A^{(1)}x = b^{(1)}$ and $A^{(2)}x = b^{(2)}$

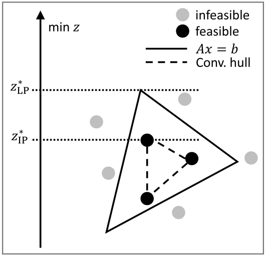
Convexification

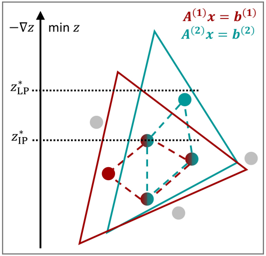
Let the LP-polytope originate from two groups of constraints $A^{(1)}x = b^{(1)}$ and $A^{(2)}x = b^{(2)}$



Convexification

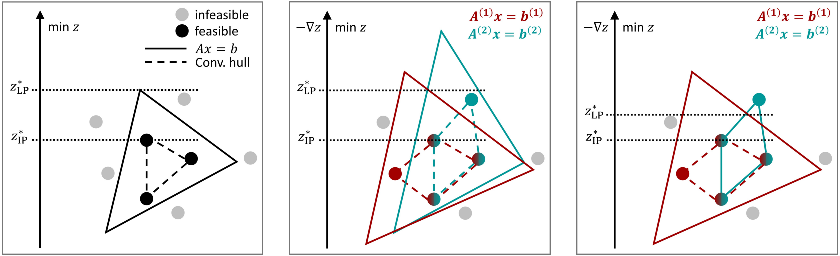
Let the LP-polytope originate from two groups of constraints $A^{(1)}x = b^{(1)}$ and $A^{(2)}x = b^{(2)}$





Convexification

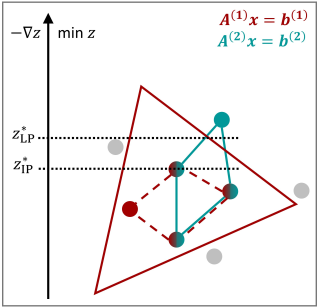
Let the LP-polytope originate from two groups of constraints $A^{(1)}x = b^{(1)}$ and $A^{(2)}x = b^{(2)}$



Knowing the convex hull wrt one group may improve strength

The reformulation [Skipping some math steps and details]

One way to know the convex hull wrt $A^{(2)}x = b^{(2)}$, $x \in \{0,1\}^n$ is to enumerate all its feasible integer solutions: a_j, $j \in \mathcal{J}$



[Since $x \in \{0,1\}^n$, the set is bounded and convexification coincides with discretisation]

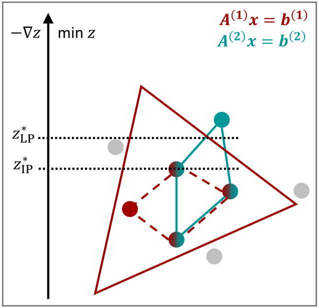
The reformulation [Skipping some math steps and details]

One way to know the convex hull wrt $A^{(2)}x = b^{(2)}$, $x \in \{0,1\}^n$ is to enumerate all its feasible integer solutions: a_j, $j \in \mathcal{J}$

For $\lambda \in \{0,1\}^{|\mathcal{J}|}$: $\sum_{j \in \mathcal{J}} \lambda_j = 1$,

solutions wrt $A^{(2)}x = b^{(2)}$, $x \in \{0,1\}^n$,

can be expressed as $x = \sum_{j \in \mathcal{J}} a_j \lambda_j$,



[Since $x \in \{0,1\}^n$, the set is bounded and convexification coincides with discretisation]

The reformulation [Skipping some math steps and details]

One way to know the convex hull wrt $A^{(2)}x = b^{(2)}$, $x \in \{0,1\}^n$ is to enumerate all its feasible integer solutions: a_j, $j \in \mathcal{J}$

For $\lambda \in \{0,1\}^{|\mathcal{J}|}$: $\sum_{j \in \mathcal{J}} \lambda_j = 1$,

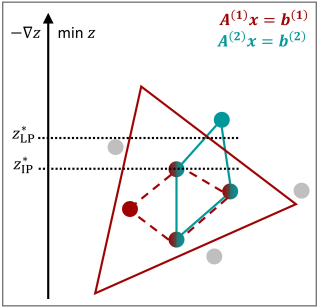
solutions wrt $A^{(2)}x = b^{(2)}$, $x \in \{0,1\}^n$,

can be expressed as $x = \sum_{j \in \mathcal{J}} a_j \lambda_j$,

and then, feasibility wrt $Ax = b$

can be expressed as

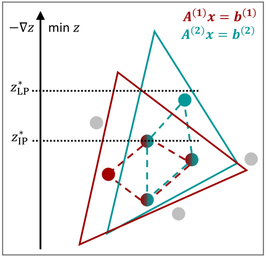
$$A^{(1)} \sum_{j \in \mathcal{J}} a_j \lambda_j = b^{(1)}$$



[Since $x \in \{0,1\}^n$, the set is bounded and convexification coincides with discretisation]

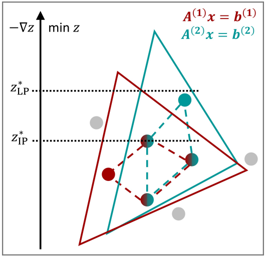
Strength of the reformulated model

Extended formulation is at least as strong as compact formulation



Strength of the reformulated model

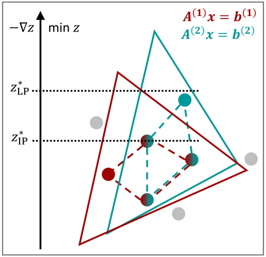
Extended formulation is at least as strong as compact formulation

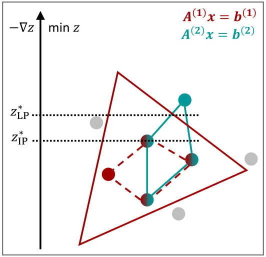


If integrality property wrt green constraints: Nothing to gain

Strength of the reformulated model

Extended formulation is at least as strong as compact formulation





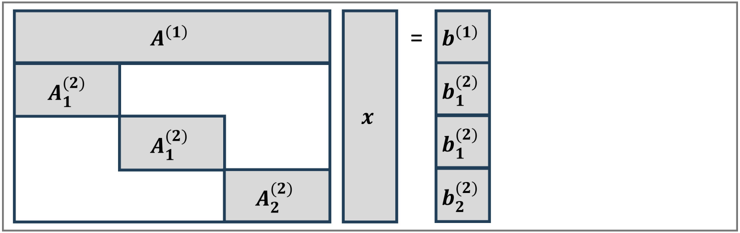
If integrality property wrt green constraints: Nothing to gain

If not integrality property wrt to the green constraints (NP-hard problem), the extended formulation might be stronger

Common type of problem structure [Several variations exists]

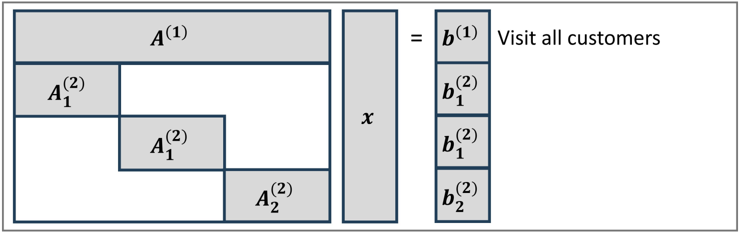
$$\begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \end{bmatrix}$$

Common type of problem structure [Several variations exists]



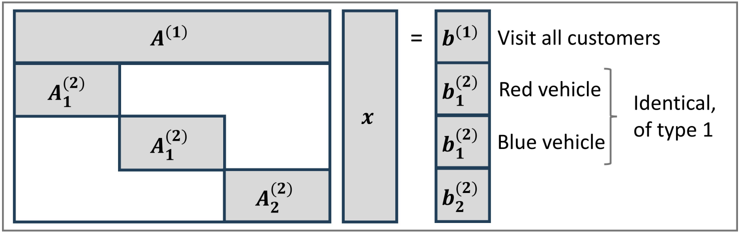
Common type of problem structure [Several variations exists]

For our vehicle routing problem



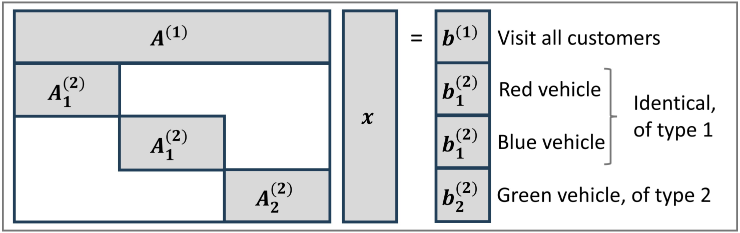
Common type of problem structure [Several variations exists]

For our vehicle routing problem



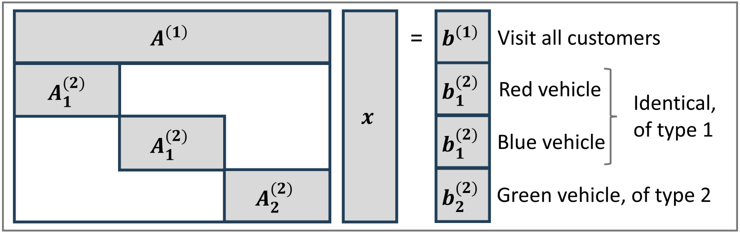
Common type of problem structure [Several variations exists]

For our vehicle routing problem



Common type of problem structure [Several variations exists]

For our vehicle routing problem



Separate enumeration of solutions for each vehicle type

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model has better properties

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes much much better properties

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes **much much stronger and structure to exploit**

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes **much much stronger and structure to exploit**

Practical impact or ”just theory”?

Air Traffic Management

Problem formulation

In the space around an airport, aircraft

- arrive at entry points in space,
- follow a path to the runway that is
- prescribed by an arrival tree

Air Traffic Management

Problem formulation

In the space around an airport, aircraft

- arrive at entry points in space,
- follow a path to the runway that is
- prescribed by an arrival tree

Design arrival tree wrt

technical requirements on descent operation, energy efficiency, collision avoidance, and complexity for air traffic controllers, ...

Air Traffic Management

Joint project

- PI Christiane Schmidt (computational geometry), Department of Science and Technology, LiU
- They are experts in modelling of routes and regulations to include all practical aspects of the problem

Swedish Research Council

Air Traffic Management

Joint project

- PI Christiane Schmidt (computational geometry), Department of Science and Technology, LiU
- They are experts in modelling of routes and regulations to include all practical aspects of the problem
- Bottleneck: Solving optimisation problem
- Postdoc project for Roghayeh Hajizadeh in my group

Swedish Research Council

Air Traffic Management

Joint project

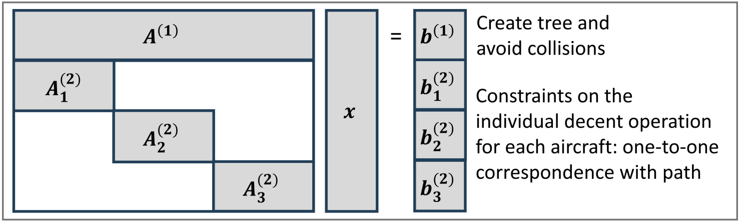
- PI Christiane Schmidt (computational geometry), Department of Science and Technology, LiU
- They are experts in modelling of routes and regulations to include all practical aspects of the problem
- Bottleneck: Solving optimisation problem
- Postdoc project for Roghayeh Hajizadeh in my group

Previous work: arc formulation over a discretisation of space. Can we do better?

Swedish Research Council

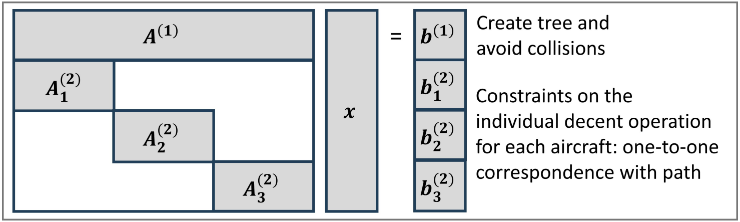
Decomposition of Air traffic management problem

It has this "common type of problem structure" ...



Decomposition of Air traffic management problem

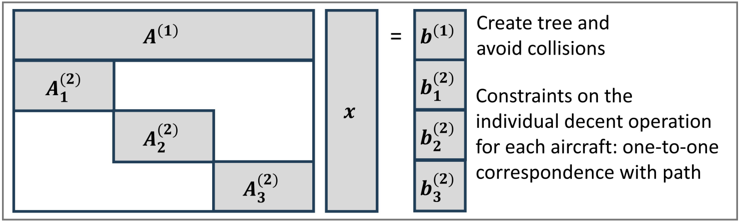
It has this "common type of problem structure" ...



... and the **possible paths are few enough to be enumerated**

Decomposition of Air traffic management problem

It has this "common type of problem structure" ...



... and the **possible paths are few enough to be enumerated**

For Arlanda runway: **Preliminary results, solution time** $\sim$ **40 hours to** $<$ **10 minutes**

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes **much much stronger and structure to exploit**

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes **much much stronger and structure to exploit**
- The number of variables increases
- **Typically the number of variables explodes** $\rightarrow$ solution space cannot be explicitly represented

Dantzig-Wolfe decomposition

A reformulation of an original compact formulation of a MIP to an extended formulation in a higher dimensional space

- New model **is at least as strong**
- Sometimes **much much stronger and structure to exploit**
- The number of variables increases
- **Typically the number of variables explodes** $\rightarrow$ solution space cannot be explicitly represented

How do we handle this?

General method idea

Extended formulation: Route $\leftrightarrow$ λ-variable $\leftrightarrow$ **column**

General method idea

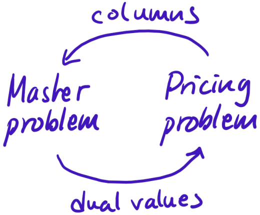
Extended formulation: Route $\leftrightarrow$ λ-variable $\leftrightarrow$ **column**

Instead of all columns:
Generate only the columns needed
for finding and verifying optimality

General method idea

Extended formulation: Route $\leftrightarrow$ λ-variable $\leftrightarrow$ **column**

Instead of all columns:
Generate only the columns needed
for finding and verifying optimality

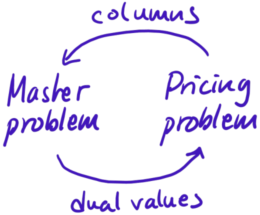


Column generation: for solving the LP relaxation
(Simplex method but find variable with negative reduced cost by solving a pricing problem = generate a column)

General method idea

Extended formulation: Route $\leftrightarrow$ λ-variable $\leftrightarrow$ **column**

Instead of all columns:
Generate only the columns needed
for finding and verifying optimality

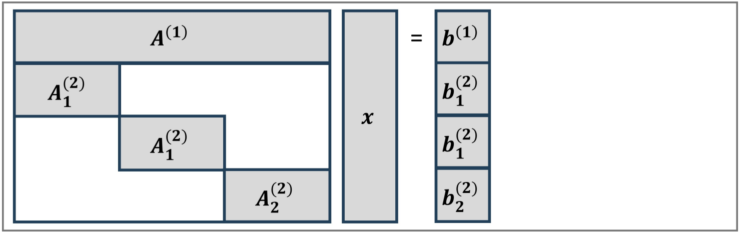


Column generation: for solving the LP relaxation
(Simplex method but find variable with negative reduced cost by solving a pricing problem = generate a column)
Branch-price-and-cut: for finding integer solutions

Notation for common structure [Several variations exists]

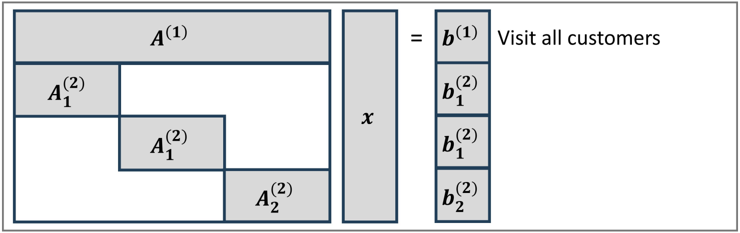
$$\begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \end{bmatrix}$$

Notation for common structure [Several variations exists]



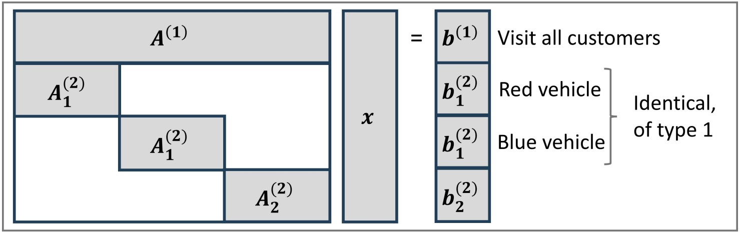
Notation for common structure [Several variations exists]

For our vehicle routing problem



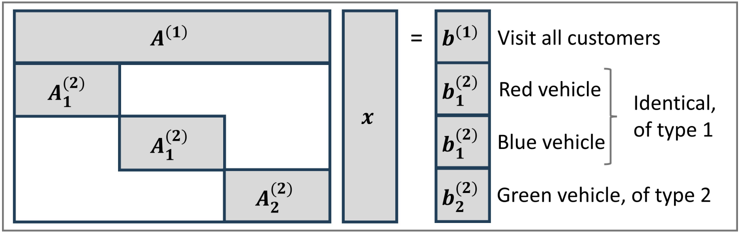
Notation for common structure [Several variations exists]

For our vehicle routing problem



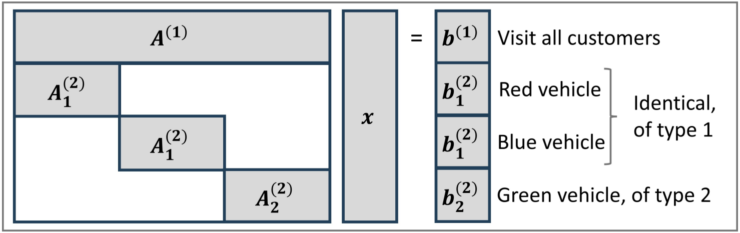
Notation for common structure [Several variations exists]

For our vehicle routing problem



Notation for common structure [Several variations exists]

For our vehicle routing problem



Set of pricing problems $Q = \{1, 2\}$ providing routes for vehicles in K_q, $q \in Q$, with $K_1 = \{\text{'Red'}, \text{'Blue'}\}$ and $K_2 = \{\text{'Green'}\}$

Models for the common structure [Skipping some math steps and details]

Master problem

Pricing problem

$$[\mathrm{CG}]_q \quad \min \quad c$$

$$\text{s.t.} \quad (c, a) \in \mathcal{A}_q$$

where

$\mathcal{A}_q$ contains feasible solutions wrt $A_q^{(2)} x = b_q^{(2)},\ x \in \{0, 1\}^n$ and their costs and

Models for the common structure [Skipping some math steps and details]

Master problem

$$\mathcal{L} = \{\lambda_j \in \{0,1\},\ j \in \mathcal{J} : \sum_{j \in \mathcal{J}_q} \lambda_j = |K_q|, q \in Q\}$$

Pricing problem

$$[\mathrm{CG}]_q \quad \min \quad c \qquad \text{s.t.} \quad (c,a) \in \mathcal{A}_q$$

where

$\mathcal{A}_q$ contains feasible solutions wrt $A_q^{(2)} x = b_q^{(2)},\ x \in \{0,1\}^n$ and their costs and

Models for the common structure [Skipping some math steps and details]

Master problem

$$
\begin{aligned}
[\mathrm{MP}] \quad \min \quad & \sum_{j \in \mathcal{J}} c_j \lambda_j, \\
\text{s.t.} \quad & A^{(1)} \sum_{j \in \mathcal{J}} a_{ij} \lambda_j = b^{(1)}, \\
& (\lambda_j)_{j \in \mathcal{J}} \in \mathcal{L} \subseteq \{0,1\}^{|\mathcal{J}|},
\end{aligned}
$$

$$
\mathcal{L} = \{\lambda_j \in \{0,1\},\ j \in \mathcal{J} : \sum_{j \in \mathcal{J}_q} \lambda_j = |K_q|, q \in Q\}
$$

Pricing problem

$$
\begin{aligned}
[\mathrm{CG}]_q \quad \min \quad & c \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q
\end{aligned}
$$

where

$\mathcal{A}_q$ contains feasible solutions wrt $A_q^{(2)} x = b_q^{(2)},\ x \in \{0,1\}^n$ and their costs and

Models for the common structure [Skipping some math steps and details]

Master problem—LP relaxation

$$
\begin{aligned}
\min \quad & \sum_{j \in \mathcal{J}} c_j \lambda_j, \\
\text{s.t.} \quad & A^{(1)} \sum_{j \in \mathcal{J}} a_{ij} \lambda_j = b^{(1)}, \\
& (\lambda_j)_{j \in \mathcal{J}} \in \mathcal{L} \subseteq [0,1]^{|\mathcal{J}|},
\end{aligned}
$$

$$
\mathcal{L} = \{\lambda_j \in [0,1],\ j \in \mathcal{J} : \sum_{j \in \mathcal{J}_q} \lambda_j = |K_q|, q \in Q\}
$$

Pricing problem

$$
\begin{aligned}
[\mathrm{CG}]_q \quad \min \quad & c - \sum_{i \in I} \bar{u}_i a_i \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q
\end{aligned}
$$

where

$\mathcal{A}_q$ contains feasible solutions wrt $A_q^{(2)} x = b_q^{(2)},\ x \in \{0,1\}^n$ and their costs and

u_i, $i \in I$, are dual variables wrt the constraints of **[MP-LP]** i.e. the LP relaxation of [MP]

Column generation: for solving the LP relaxation of [MP]

Column generation: for solving the LP relaxation of [MP]

Restricted master problem

$$[\text{MP-LP}] \quad \min \sum_{j \in J} c_j \lambda_j,$$
$$\text{s.t.} \quad A^{(1)} \sum_{j \in J} a_{ij} \lambda_j = b^{(1)},$$
$$(\lambda_j)_{j \in J} \in \mathcal{L} \subseteq [0,1]^{|J|},$$

$$\mathcal{L} = \{\lambda_j \in [0,1],\ j \in J : \sum_{j \in J_q} \lambda_j = |K_q|, q \in Q\}$$

Build *restricted master problem* with $J \subseteq \mathcal{J}$ iteratively

Column generation: for solving the LP relaxation of [MP]

Restricted master problem

$$
\text{[MP-LP]} \quad \min \sum_{j \in J} c_j \lambda_j,
$$
$$
\text{s.t.} \quad A^{(1)} \sum_{j \in J} a_{ij} \lambda_j = b^{(1)},
$$
$$
(\lambda_j)_{j \in J} \in \mathcal{L} \subseteq [0,1]^{|J|},
$$

$$
\mathcal{L} = \{\lambda_j \in [0,1],\ j \in J : \sum_{j \in J_q} \lambda_j = |K_q|, q \in Q\}
$$

Build *restricted master problem* with $J \subseteq \mathcal{J}$ iteratively

- Add λ-variable with minimum reduced cost: pivot into the basis $\Leftrightarrow$ simplex-method iteration

Column generation: for solving the LP relaxation of [MP]

Restricted master problem

$$
\text{[MP-LP]} \quad \min \sum_{j \in J} c_j \lambda_j,
$$
$$
\text{s.t.} \quad A^{(1)} \sum_{j \in J} a_{ij} \lambda_j = b^{(1)},
$$
$$
(\lambda_j)_{j \in J} \in \mathcal{L} \subseteq [0,1]^{|J|},
$$

$$
\mathcal{L} = \{\lambda_j \in [0,1],\ j \in J : \sum_{j \in J_q} \lambda_j = |K_q|, q \in Q\}
$$

Build *restricted master problem* with $J \subseteq \mathcal{J}$ iteratively

- Add λ-variable with minimum reduced cost: pivot into the basis $\Leftrightarrow$ simplex-method iteration
- Negative reduced cost sufficient for improvement
- Stop when no negative reduced cost is returned

Column generation: integer solutions?

- LP column generation:
  Generated subspace is sufficient for solving the LP relaxation
- It may or may not include high-quality integer solutions

Column generation: integer solutions?

- LP column generation:
  Generated subspace is sufficient for solving the LP relaxation
- It may or may not include high-quality integer solutions

- Restricted master heuristic / price-and-branch:
  solve an integer program over this subspace

Column generation: integer solutions?

- LP column generation:
  Generated subspace is sufficient for solving the LP relaxation
- It may or may not include high-quality integer solutions

- Restricted master heuristic / price-and-branch:
  solve an integer program over this subspace
- To obtain integer optimality:
  - Perform branching and add cuts
  - Generate columns for LP relaxations involved

  $\rightarrow$ **Branch-price-and-cut**

Branch-price-and-cut

Relies on what is known from branching and cutting in MIP—
but adaptations are required and caution is advised

Branch-price-and-cut

Relies on what is known from branching and cutting in MIP—
but adaptations are required and caution is advised

- Complete solution space not available
- Need to "play well" with both [MP] and pricing

Branch-price-and-cut

Relies on what is known from branching and cutting in MIP—
but adaptations are required and caution is advised

- Complete solution space not available
- Need to ”play well” with both [MP] and pricing
- Common with customised branching schemes and cuts

Branch-price-and-cut

Relies on what is known from branching and cutting in MIP—
but adaptations are required and caution is advised

- Complete solution space not available
- Need to "play well" with both [MP] and pricing
- Common with customised branching schemes and cuts

Extensive literature and knowledge, often problem specific

Branch-price-and-cut

Relies on what is known from branching and cutting in MIP—
but adaptations are required and caution is advised

- Complete solution space not available
- Need to "play well" with both [MP] and pricing
- Common with customised branching schemes and cuts

Extensive literature and knowledge, often problem specific

No time for details today: let's zoom in on a specific topic ...

Optimality conditions

LP column generation: Follows directly from LP theory

Restricted master problem solved to optimality &
no negative reduced costs found in pricing

Optimality conditions

LP column generation: Follows directly from LP theory

Restricted master problem solved to optimality &
no negative reduced costs found in pricing

Subspace sufficient for solving the integer program?

Some answers, but there is more to be understood

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

E. Rönnberg and T. Larsson. *An integer optimality condition for column generation on zero-one linear programs*. Discrete Optimization, 31:79–92, 2019.

Heuristics—based on LP pricing

Possible to apply any heuristic on the restricted master problem–
BUT this limits you to the solutions in the generated subspace

Heuristics—based on LP pricing

Possible to apply any heuristic on the restricted master problem–
BUT this limits you to the solutions in the generated subspace

Beyond that, e.g diving heuristics, feasibility pump, crossover, ...

R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, and E. Uchoa. *Primal heuristics for branch and price: The assets of diving methods.* INFORMS Journal on Computing, 31(2):251–267, 2019.

P. Pesneau, R. Sadykov, and F. Vanderbeck. *Feasibility pump heuristics for column generation approaches.* In International Symposium on Experimental Algorithms, pages 332–343. Springer, 2012.

M. Lübbecke and C. Puchert. *Primal heuristics for branch-and-price algorithms.* In Operations Research Proceedings 2011, pages 65–70. Springer, 2012.

Heuristics—pricing for integrality

Use the quasi-integrality property [also as exact method]

- Initial contributions by E. Rönnberg and T. Larsson, $2\times$EJOR
- Much more mature line of work by the Montreal group, including F. Soumis, I. El Hallaoui, G. Desaulniers, ...

Heuristics—pricing for integrality

Use the quasi-integrality property [also as exact method]

- Initial contributions by E. Rönnberg and T. Larsson, 2×EJOR
- Much more mature line of work by the Montreal group, including F. Soumis, I. El Hallaoui, G. Desaulniers, ...

In a more general sense:
Is it possible to directly generate columns that make the restricted master problem include improved integer solutions?
Can we price for integrality?

Large Neighbourhood Search (LNS) heuristics

Important component in branch-and-bound-based MIP solvers (diving, feasibility pump, local branching, ...)

- Solve an auxiliary problem to find an improved integer solution
- Also known as sub-MIPing

Large Neighbourhood Search (LNS) heuristics

Important component in branch-and-bound-based MIP solvers (diving, feasibility pump, local branching, ...)

- Solve an auxiliary problem to find an improved integer solution
- Also known as sub-MIPing

LNS heuristics & branch-price-and-cut?

- Destroy method: Remove columns from a current solution
- Repair method: Generate new useful ones to complement

Large Neighbourhood Search (LNS) heuristics

Important component in branch-and-bound-based MIP solvers (diving, feasibility pump, local branching, ...)

- Solve an auxiliary problem to find an improved integer solution
- Also known as sub-MIPing

LNS heuristics & branch-price-and-cut?

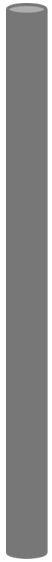
- Destroy method: Remove columns from a current solution
- Repair method: Generate new useful ones to complement

As before: "Adaptation is required and caution is advised"

Can we make an LNS price for integrality?

Illustrations and VRP interpretations

Column $=$ binary vector $(a_{ij})_{i \in I}$



$$= \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Corresponds to a route and indicates if customer i is visited by the vehicle or not

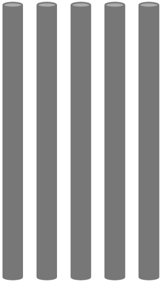
Illustrations and VRP interpretations

Column $=$ binary vector $(a_{ij})_{i \in I}$

$$= \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Corresponds to a route and indicates if customer i is visited by the vehicle or not

Example: **feasible solution**



$$= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

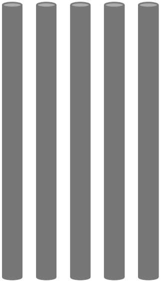
5 routes that together visit each costumer exactly once

Illustrations and VRP interpretations

Column $=$ binary vector $(a_{ij})_{i \in I}$

$$= \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Example: **feasible solution**



$$= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Decision variables:

$$\lambda_j = \begin{cases} 1 \text{ if column } j \in \mathcal{J}_q \text{ of pricing problem } q \in Q \text{ is used,} \\ 0 \text{ otherwise} \end{cases}$$

Notation

$$
\text{[MP]} \quad \min \sum_{j \in \mathcal{J}} c_j \lambda_j,
$$

$$
\text{s.t.} \quad \sum_{j \in \mathcal{J}} a_{ij} \lambda_j \geq 1, \quad i \in I^{\mathrm{c}},
$$

$$
\sum_{j \in \mathcal{J}} a_{ij} \lambda_j \leq 1, \quad i \in I^{\mathrm{p}},
$$

$$
(\lambda_j)_{j \in \mathcal{J}} \in \mathcal{L} \subseteq \{0,1\}^{|\mathcal{J}|},
$$

$$
\mathcal{L} = \{\lambda_j \in \{0,1\}, j \in \mathcal{J} : \sum_{j \in \mathcal{J}_q} \lambda_j = |K_q|, q \in Q\}.
$$

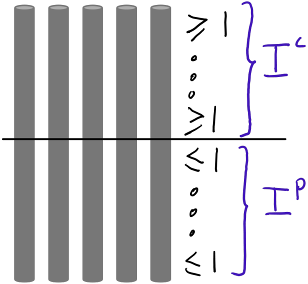
Notation

$$[\mathrm{MP}] \quad \min \sum_{j \in \mathcal{J}} c_j \lambda_j,$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} a_{ij} \lambda_j \geq 1, \quad i \in I^{\mathrm{c}},$$

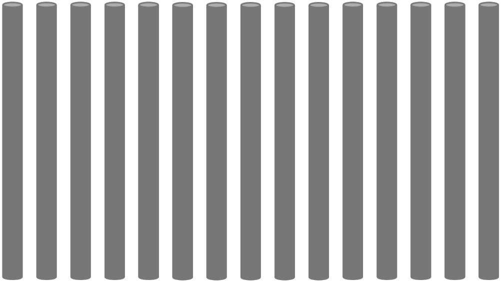
$$\sum_{j \in \mathcal{J}} a_{ij} \lambda_j \leq 1, \quad i \in I^{\mathrm{p}},$$

$$(\lambda_j)_{j \in \mathcal{J}} \in \mathcal{L} \subseteq \{0, 1\}^{|\mathcal{J}|},$$



LNS – Destroy method

Columns in RMP:
$J_q,\ q \in Q$

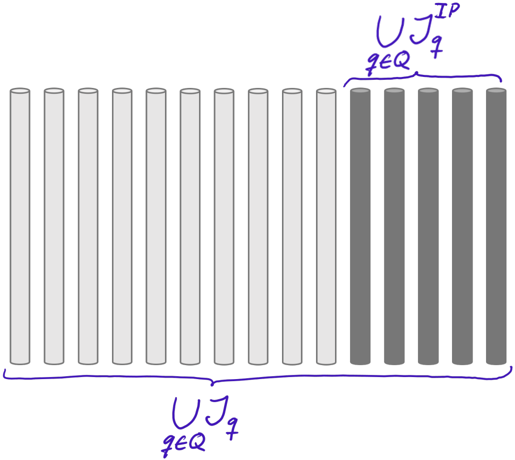


$\bigcup_{q \in Q} J_q$

LNS – Destroy method

Columns in RMP:
$J_q,\ q \in Q$

Current solution = active columns:
$J_q^{\mathrm{IP}}, q \in Q$



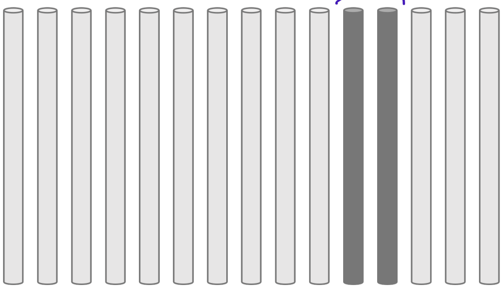
LNS – Destroy method

Columns in RMP:
$J_q,\ q \in Q$

Current solution = active columns:
$J_q^{\mathrm{IP}}, q \in Q$

Destroy method = Remove active columns

$\hat{J} = \bigcup_{q \in Q} \hat{J}_q$



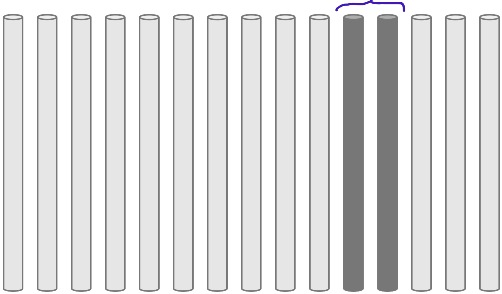
LNS – Destroy method

Columns in RMP:
$J_q,\ q \in Q$

Current solution = active columns:
$J_q^{\mathrm{IP}}, q \in Q$

Destroy method = Remove active columns

$\hat{J} = \bigcup_{q \in Q} \hat{J}_q$



Let the set of remaining columns $\hat{J}$ be fixed:
What is the best possible way to repair the solution?

LNS – "Ideal" repair method

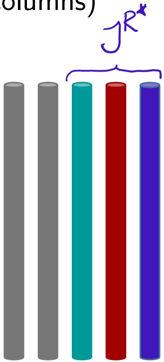
Solve [REP] over the set $J^{\mathrm{R}} = \mathcal{J}$ (all possible columns)

$$
\begin{aligned}
\text{[REP]} \quad \min \quad & \sum_{j \in J^{\mathbf{R}}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \geq 1 - \sum_{j \in \hat{J}} a_{ij}, \ i \in I^{\mathrm{c}}, \\
& \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \leq 1 - \sum_{j \in \hat{J}} a_{ij}, \ i \in I^{\mathrm{p}}, \\
& \sum_{j \in J_q^{\mathbf{R}}} \lambda_j = |K_q| - |\hat{J}_q|, \ q \in Q, \\
& \lambda_j \in \{0, 1\}, j \in J^{\mathrm{R}} \cup J.
\end{aligned}
$$

LNS – "Ideal" repair method

Solve [REP] over the set $J^{\mathrm{R}} = \mathcal{J}$ (all possible columns)

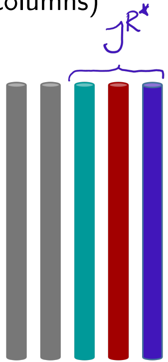
$$
\begin{aligned}
\text{[REP]} \qquad \min \quad & \sum_{j \in J^{\mathbf{R}}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \geq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{c}}, \\
& \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \leq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{p}}, \\
& \sum_{j \in J^{\mathbf{R}}_q} \lambda_j = |K_q| - |\hat{J}_q|, \quad q \in Q, \\
& \lambda_j \in \{0,1\}, j \in J^{\mathrm{R}} \cup J.
\end{aligned}
$$



LNS – "Ideal" repair method

Solve [REP] over the set $J^{\mathrm{R}} = \mathcal{J}$ (all possible columns)

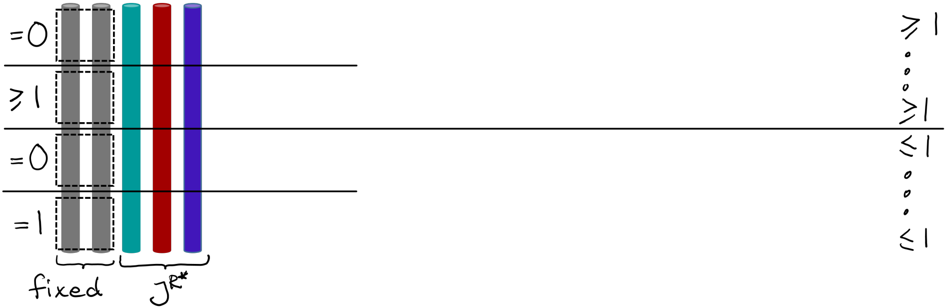
$$
\begin{aligned}
\text{[REP]} \qquad \min \quad & \sum_{j \in J^{\mathbf{R}}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \geq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{c}}, \\
& \sum_{j \in J^{\mathbf{R}}} a_{ij} \lambda_j \leq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{p}}, \\
& \sum_{j \in J_q^{\mathbf{R}}} \lambda_j = |K_q| - |\hat{J}_q|, \quad q \in Q, \\
& \lambda_j \in \{0, 1\}, j \in J^{\mathrm{R}} \cup J.
\end{aligned}
$$



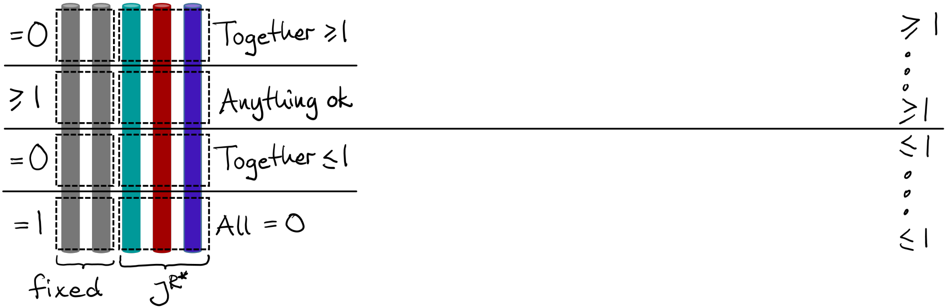
NOT reasonable in practice!

Properties of J^{R^*} and desired properties of J^R

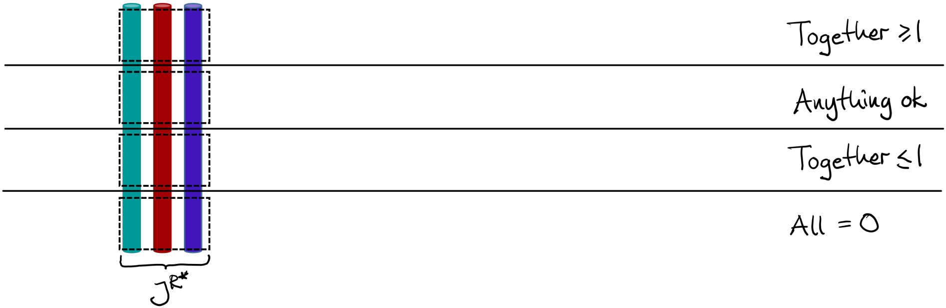
Properties of J^{R^*} and desired properties of J^{R}



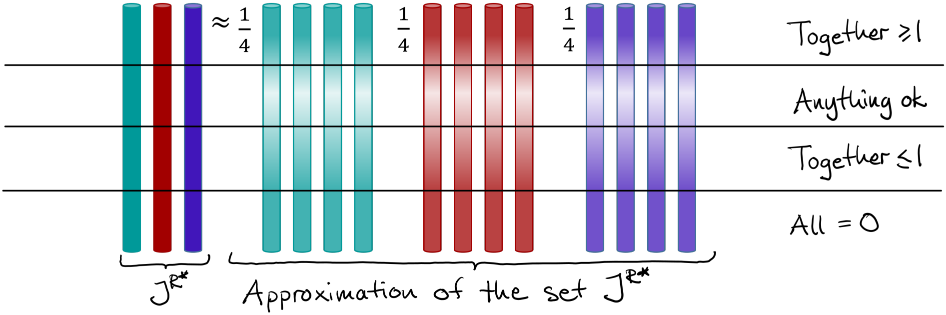
Properties of J^{R^*} and desired properties of J^R



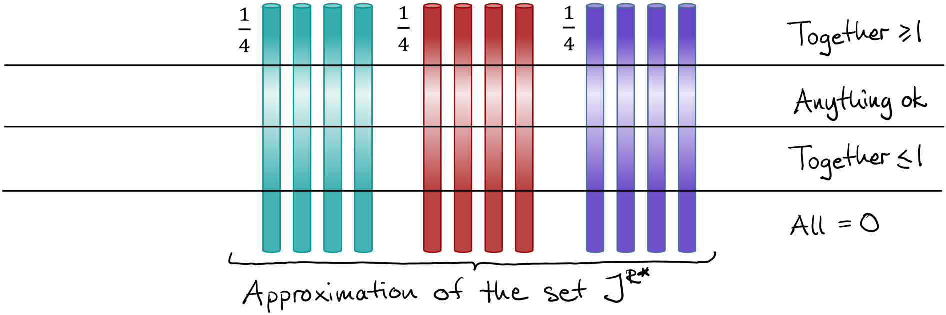
Properties of J^{R^*} and desired properties of J^R



Properties of J^{R^*} and desired properties of J^R



Properties of J^{R^*} and desired properties of J^R



→ **Aim for these properties when generating J^R**

Desired properties translated to the pricing problem

- "Anything ok" $\Rightarrow$ no change in the pricing problem

Desired properties translated to the pricing problem

- "Anything ok" $\Rightarrow$ no change in the pricing problem
- "All $= 0$" $\Rightarrow$ Big-M penalty on corresponding a_i

Desired properties translated to the pricing problem

- "Anything ok" $\Rightarrow$ no change in the pricing problem
- "All $= 0$" $\Rightarrow$ Big-M penalty on corresponding a_i
- "Together ≥ 1 or ≤ 1" $\Rightarrow$

Desired properties translated to the pricing problem

- "Anything ok" $\Rightarrow$ no change in the pricing problem
- "All = 0" $\Rightarrow$ Big-M penalty on corresponding a_i
- "Together ≥ 1 or ≤ 1" $\Rightarrow$
  In iteration l, aim at complying with

$$\sum_{j \in J^{R*}} \sum_{j' \in \hat{L}_{jl}} a_{ij'} \begin{cases} \geq \frac{1}{|J^{R*}|} \sum_{j \in J^{R*}} |\hat{L}_{jl}|, & i \in \hat{I}^{c0}, \\ \leq \frac{1}{|J^{R*}|} \sum_{j \in J^{R*}} |\hat{L}_{jl}|, & i \in \hat{I}^{p0}. \end{cases}$$

Desired properties translated to the pricing problem

- "Anything ok" $\Rightarrow$ no change in the pricing problem
- "All = 0" $\Rightarrow$ Big-M penalty on corresponding a_i
- "Together ≥ 1 or ≤ 1" $\Rightarrow$
  In iteration l, aim at complying with

$$\sum_{j \in J^{R*}} \sum_{j' \in \hat{L}_{jl}} a_{ij'} \begin{cases} \geq \frac{1}{|J^{R*}|} \sum_{j \in J^{R*}} |\hat{L}_{jl}|, & i \in \hat{I}^{c0}, \\ \leq \frac{1}{|J^{R*}|} \sum_{j \in J^{R*}} |\hat{L}_{jl}|, & i \in \hat{I}^{p0}. \end{cases}$$

Just simple calculations and comparisons in each iteration – adjust penalties on the corresponding a_i:s dynamically

LINKÖPING UNIVERSITY

Repair pricing

Pricing problem q in iteration l

$$[\text{REP-CG}_{ql}] \quad \min \quad c - \sum_{i \in I^{\mathsf{c}}} \bar{u}_i a_i + \sum_{i \in I^{\mathsf{p}}} \bar{u}_i a_i$$

$$\text{s.t.} \quad (c, a) \in \mathcal{A}_q.$$

Repair pricing

Pricing problem q in iteration l

$$
\begin{aligned}
[\text{REP-CG}_{ql}] \quad \min \quad & c - \sum_{i \in I^{\mathrm{c}}} \bar{u}_i a_i + \sum_{i \in I^{\mathrm{p}}} \bar{u}_i a_i + \\
& + \sum_{i \in \hat{I}^{\mathrm{p1}}} M a_i - \sum_{i \in \hat{I}^{\mathrm{c0}}} \beta_{il} a_i + \sum_{i \in \hat{I}^{\mathrm{p0}}} \beta_{il} a_i \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q.
\end{aligned}
$$

- Static Big-M penalties and dynamic penalties β_{il}

Repair pricing

Pricing problem q in iteration l

$$
\begin{aligned}
[\text{REP-CG}_{ql}] \quad \min \quad & c - \sum_{i \in I^{\mathrm{c}}} \gamma \bar{u}_i a_i + \sum_{i \in I^{\mathrm{p}}} \gamma \bar{u}_i a_i + \\
& + \sum_{i \in \hat{I}^{\mathrm{p1}}} M a_i - \sum_{i \in \hat{I}^{\mathrm{c0}}} \beta_{il} a_i + \sum_{i \in \hat{I}^{\mathrm{p0}}} \beta_{il} a_i \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q .
\end{aligned}
$$

- Static Big-M penalties and dynamic penalties β_{il}
- Adjust the reduced costs with the parameter $\gamma \in [0, 1]$ **to heuristically price for integrality**

Repair pricing

Pricing problem q in iteration l

$$
\begin{aligned}
[\text{REP-CG}_{ql}] \quad \min \quad & c - \sum_{i \in I^{\mathrm{c}}} \gamma \bar{u}_i a_i + \sum_{i \in I^{\mathrm{p}}} \gamma \bar{u}_i a_i + \\
& + \sum_{i \in \hat{I}^{\mathrm{p1}}} M a_i - \sum_{i \in \hat{I}^{\mathrm{c0}}} \beta_{il} a_i + \sum_{i \in \hat{I}^{\mathrm{p0}}} \beta_{il} a_i \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q.
\end{aligned}
$$

- Static Big-M penalties and dynamic penalties β_{il}
- Adjust the reduced costs with the parameter $\gamma \in [0, 1]$ **to heuristically price for integrality—why?**

In pursuit of γ: Detour via Lagrangian relaxation

$$
\begin{aligned}
z^* \;=\; \min \quad & \sum_{j \in \mathcal{J}} c_j x_j \\
\text{s.t.} \quad & \sum_{j \in \mathcal{J}} A_j x_j \geq b \\
& x_j \in \{0,1\}, \; j \in \mathcal{J}
\end{aligned}
$$

Lagrangian function:

$$L(x,u) = \sum_{j \in \mathcal{J}} c_j x_j + u^{\mathsf{T}} \left(b - \sum_{j \in \mathcal{J}} A_j x_j \right)$$

Lagrangian dual function:

$h(u) = \min_x L(x,u)$

Duality gap:

$\Gamma = z^* - h^*, \;$ with $h^* = \max_u h(u)$

In pursuit of γ: Lagrangian relaxation—optimality conditions

Equivalent statements:

- x solves the primal problem
  u solves the dual problem
  the duality gap $\Gamma = 0$

- Lagrangian optimality: $L(x,u) \leq h(u)$
  Primal feasibility: $\sum_{j\in\mathcal{J}} A_j x_j \geq b$
  Complementarity: $u^{\mathsf{T}}\left(b - \sum_{j\in\mathcal{J}} A_j x_j\right) = 0$

In pursuit of γ: Lagrangian relaxation—discrete problems

Optimality conditions are for problems with no duality gap:
But discrete problems typically have a positive duality gap

In pursuit of γ: Lagrangian relaxation—discrete problems

Optimality conditions are for problems with no duality gap:
But discrete problems typically have a positive duality gap

Use generalised optimality conditions by Larsson and Patriksson:
[T. Larsson, M. Patriksson. *Global optimality conditions for discrete and nonconvex optimization – with applications to Lagrangian heuristics and column generation.* Operations Research (2006)]

For a binary x and a $u \geq 0$ introduce:

- ε-optimality in the Lagrangian problem
$$\boldsymbol{\varepsilon}(x, u) = u^{\mathsf{T}} b + \sum_{j \in \mathcal{J}} \left(c_j - u^{\mathsf{T}} A_j\right) x_j - h(u)$$
- δ-complementarity
$$\boldsymbol{\delta}(x, u) = u^{\mathsf{T}} \left(\sum_{j \in \mathcal{J}} A_j x_j - b \right)$$

In pursuit of γ: Optimality conditions—discrete problems

Equivalent statements:

- x solves the primal problem and u solves the dual problem

In pursuit of γ: Optimality conditions—discrete problems

Equivalent statements:

- x solves the primal problem and u solves the dual problem
- Lagrangian optimality: $L(x,u) \leq h(u) + \varepsilon(x,u)$

  Primal feasibility: $\sum_{j \in \mathcal{J}} A_j x_j \geq b$

  Complementarity: $u^{\mathsf{T}} \left(b - \sum_{j \in \mathcal{J}} A_j x_j \right) \geq -\delta(x,u)$

  $\varepsilon(x,u) + \delta(x,u) \leq \Gamma$, and $\varepsilon(x,u), \delta(x,u) \geq 0$

In pursuit of γ: Pricing with respect to ε and $\boldsymbol{\delta}$

- Traditional pricing = minimise wrt ε
- Optimality conditions suggest minimising wrt ε **and** $\boldsymbol{\delta}$

New column wrt minimising $\alpha\varepsilon + (1-\alpha)\,\delta,\ \alpha \in [0, 1/2] \ \Leftrightarrow$

$$\min_{j \in \mathcal{J}} c_j - \gamma u^{\mathsf{T}} A_j,\ \gamma \in [0, 1]$$

[Y. Zhao, T. Larsson, E. Rönnberg. *An integer programming column generation principle for heuristic search methods.* International Transactions in Operational Research, 27:665–695, 2020.]

Heuristic pricing for integrality

LNS heuristics of destroy-repair type

- Destroy method: Remove columns from a current solution
- Repair method: Generate a set of columns "with profitable properties"

Heuristic pricing for integrality

LNS heuristics of destroy-repair type

- Destroy method: Remove columns from a current solution
- Repair method: Generate a set of columns "with profitable properties"

Two implementations

- IPColGen as part of the B&P&C scheme in GCG (SCIP)
  [S. J. Maher and E. Rönnberg. Integer programming column generation: accelerating branch-and-price using ...
  Mathemathical Programming Computation, (15):509–548, 2023.]

Heuristic pricing for integrality

LNS heuristics of destroy-repair type

- Destroy method: Remove columns from a current solution
- Repair method: Generate a set of columns "with profitable properties"

Two implementations

- IPColGen as part of the B&P&C scheme in GCG (SCIP)
  [S. J. Maher and E. Rönnberg. Integer programming column generation: accelerating branch-and-price using ...
  Mathemathical Programming Computation, (15):509–548, 2023.]
- Problem-specific implementation for an EVRP

IPColGen in GCG module of SCIP

Implemented as part of the B&P&C scheme in GCG

- Apply in root node

IPColGen in GCG module of SCIP

Implemented as part of the B&P&C scheme in GCG

- Apply in root node when
  - when tailing-off for the LP-relaxation begins

IPColGen in GCG module of SCIP

Implemented as part of the B&P&C scheme in GCG

- Apply in root node when
  - when tailing-off for the LP-relaxation begins
  - optimality gap is large (= expected to be of most use)

IPColGen in GCG module of SCIP

Implemented as part of the B&P&C scheme in GCG

- Apply in root node when
  - when tailing-off for the LP-relaxation begins
  - optimality gap is large (= expected to be of most use)
- Apply for a subset of the nodes in the B&P tree
  (too expensive to use in all nodes)

IPColGen in GCG module of SCIP

Implemented as part of the B&P&C scheme in GCG

- Apply in root node when
  - when tailing-off for the LP-relaxation begins
  - optimality gap is large (= expected to be of most use)
- Apply for a subset of the nodes in the B&P tree (too expensive to use in all nodes)

Evaluated when used in addition to all other heuristics in GCG/SCIP to compare to its state of the art

Evaluation measures

- All results as a function of first call gap

Evaluation measures

- All results as a function of first call gap
- Primal integral
  - Common way to measure progress of heuristics
  - Each point in time: integral over primal gap as function of time
- Primal / optimality gap after 3,600s

Evaluation measures

- All results as a function of first call gap
- Primal integral
  - Common way to measure progress of heuristics
  - Each point in time: integral over primal gap as function of time
- Primal / optimality gap after 3,600s
- Diverse test set:
  Shifted geometric mean
- Display ratio with/without IPColGen

Evaluation measures

- All results as a function of first call gap
- Primal integral
  - Common way to measure progress of heuristics
  - Each point in time: integral over primal gap as function of time
- Primal / optimality gap after 3,600s
- Diverse test set: Shifted geometric mean
- Display ratio with/without IPColGen

Essentially: A value < 1 means we perform well

Instances with known block diagonal structures

Results for about 700 instances

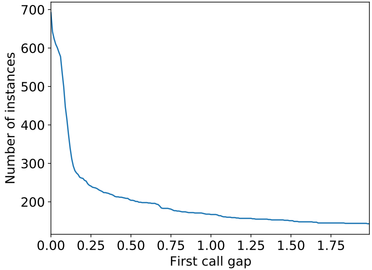
- Bin packing
- Capacitated p-median
- Generalised assignment
- Vertex coloring
- Optimal interval scheduling

Instances with known block diagonal structures

Results for about 700 instances

- Bin packing
- Capacitated p-median
- Generalised assignment
- Vertex coloring
- Optimal interval scheduling

Instance characteristics

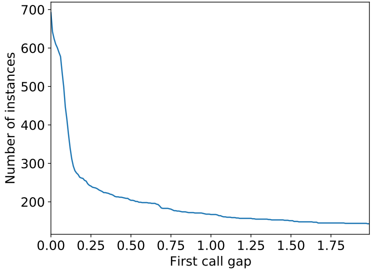


Instances with known block diagonal structures

Results for about 700 instances

- Bin packing
- Capacitated p-median
- Generalised assignment
- Vertex coloring
- Optimal interval scheduling

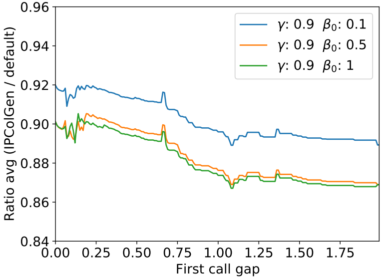
Instance characteristics



Show results for some parameter settings γ and β

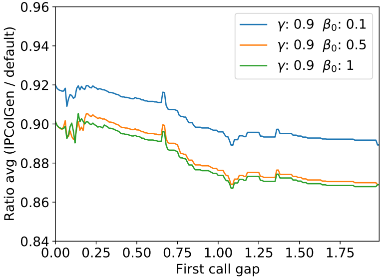
Results: Instances with known block diagonal structures

Final optimality gap

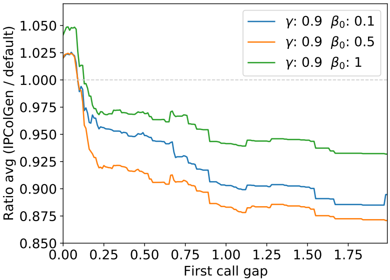


Results: Instances with known block diagonal structures

Final optimality gap

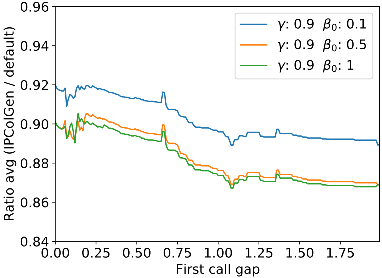


Primal integral

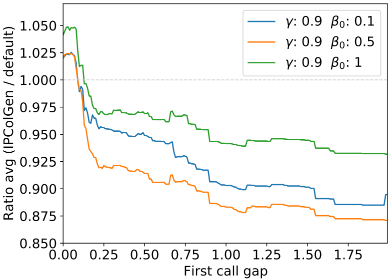


Results: Instances with known block diagonal structures

Final optimality gap



Primal integral



- better primal solutions + better final gap for all instances
- better primal integral only for instances with large initial gap

Instances from MIPLIB 2017

Results for about 160 instances with known solution and tags

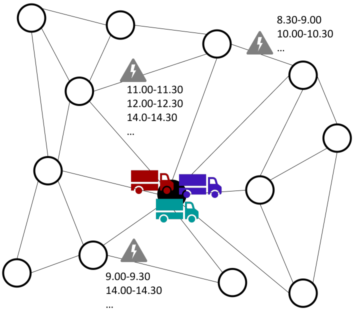
- Decomposition
- Set covering
- Set packing
- Set partitioning

Automatic structure detection & D-W decomposition in GCG:

Same type of results as for the structured instances

EVRPTW with Charging Time Slots

- Homogenous vehicles
  - Capacity
  - Linear charging rate
- Customers
  - Capacity
  - Service time
  - Time window
- Bookable charging slots



PhD student Lukas Eveborn
Preliminary results at VeRoLog2025

SCANIA FFI Fordonsstrategisk Forskning och Innovation

EVRPTW with Charging Time Slots

Part of customised implementation in GCG:

Primal heuristics
- RMP-heuristic
- **Heuristic pricing for integrality**

GCG[1]/SCIP [2]
- Pricing coordination
- Some heuristics
- Branching
- Stabilization

Labelling Algorithm [3]
- ng-routes [4]
- graph-reduction
- Resource bounds
- Completions bounds

Customised cuts
- 2-path-cuts [5]

Customised branching
- Number of vehicles

[1] G. Gamrath, M. Lübbecke (2010), [2] S. Bolusani et. al (2024). [3] J. Enerbäck, L. Eveborn, E. Rönnberg (2024).
[4] R. Baldacci, A. Mingozzi, R. Roberti (2011). [5] N. Kohl et. al (1999).

Heuristic pricing for integrality closes 1/3 of root node gap

Concluding comments

Branch-price-and-cut relies on LP-pricing to find a subspace that contains an optimal integer solution.

Room for improvements?

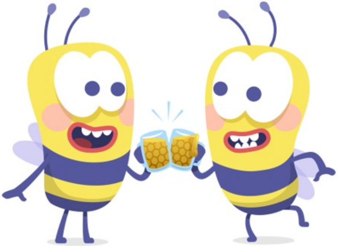
- Optimality conditions
- Pricing for integrality

Today:
Some contributions in this direction—but more to be understood!

Final notes ...

Acknowledgements:

- The Center for Industrial Information Technology (CENIIT)
- Swedish National Infrastructure for Computing (SNIC) and National Academic Infrastructure for Supercomputing in Sweden (NAISS)
- FFI
- Scania



Thanks for listening!