Integer programming column generation:

Accelerating branch-and-price for set covering, packing, and partitioning problems

Stephen J. Maher and Elina Rönnberg

Column generation 2023

LINKÖPING UNIVERSITY

What?

- A Large Neighbourhood Search (LNS) heuristic for extended formulations
- Implemented in GCG: A generic branch-price-and-cut solver in SCIP
- Improves computational performance of GCG 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
- "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:
  Solve a sub-MIP using columns from a specialised repair pricing scheme

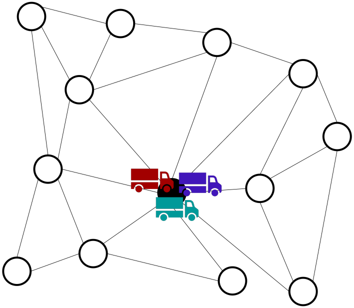
Key contribution: The repair pricing scheme

Outline

Running example: Model for VRP

Problem formulation

Use these three vehicles
Visit all customers
Minimise total travel time

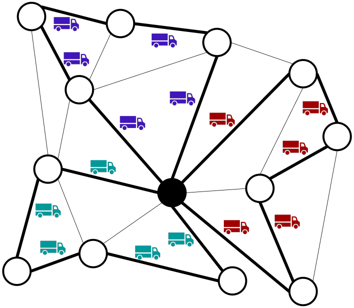


Running example: Model for VRP

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

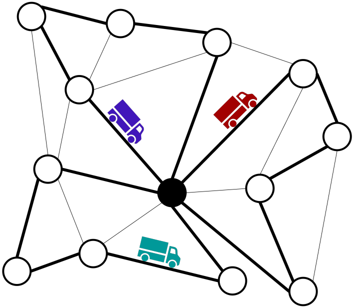
Running example: Model for VRP

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



Running example: Model for VRP

Extended formulation

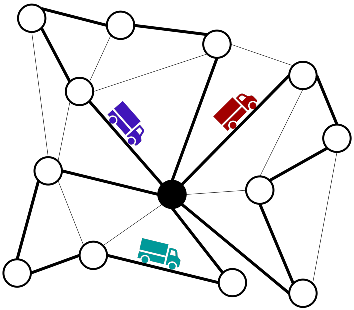
Decision variables:

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

Constraints:

One route per vehicle

Vehicles cover all customers

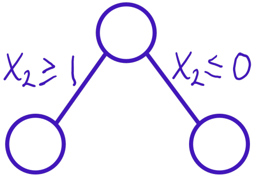


Feasible routes are constructed by solving a pricing problem

Branching for different formulations

Compact formulation

$x=(1,1/2)$



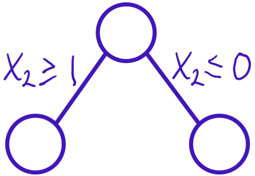
$x_2 \geq 1$: Force vehicle to use arc

$x_2 \leq 0$: Forbid vehicle to use arc

Branching for different formulations

Compact formulation

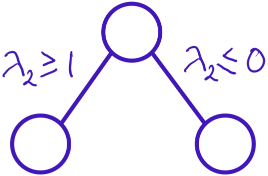
$x = (1, 1/2)$



$x_2 \geq 1$: Force vehicle to use arc
$x_2 \leq 0$: Forbid vehicle to use arc

Extended formulation

$\lambda = (1, 1/2)$

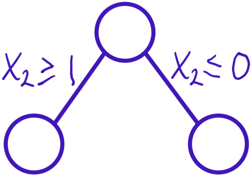


$\lambda_2 \geq 1$: Force use of route
$\lambda_2 \leq 0$: Forbid use of route

Branching for different formulations

Compact formulation

$x = (1, 1/2)$

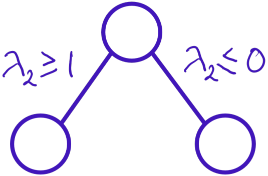


$x_2 \geq 1$: Force vehicle to use arc

$x_2 \leq 0$: Forbid vehicle to use arc

Extended formulation

$\lambda = (1, 1/2)$



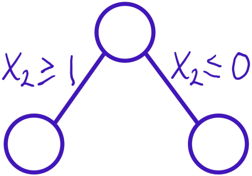
$\lambda_2 \geq 1$: Force use of route

$\lambda_2 \leq 0$: Forbid use of route & **never generate it again**

Branching for different formulations

Compact formulation

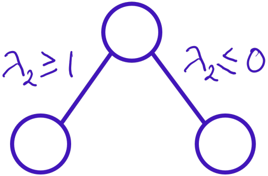
$x = (1, 1/2)$



$x_2 \geq 1$: Force vehicle to use arc
$x_2 \leq 0$: Forbid vehicle to use arc

Extended formulation

$\lambda = (1, 1/2)$



$\lambda_2 \geq 1$: Force use of route
$\lambda_2 \leq 0$: Forbid use of route & **never generate it again**

Problem: No computationally efficient way to prevent one exact route/column/solution from being generated

Well-known challenge: Branching in branch-and-price

Instead of the ”naïve” branching $\lambda_2 \geq 1$ and $\lambda_2 \leq 0$:

- Branch on variables of the corresponding compact formulation
- Translates to using or omitting one arc in the pricing problem

Common with customised branching schemes to achieve this

Well-known challenge: Branching in branch-and-price

Instead of the ”naïve” branching $\lambda_2 \geq 1$ and $\lambda_2 \leq 0$:

- Branch on variables of the corresponding compact formulation
- Translates to using or omitting one arc in the pricing problem

Common with customised branching schemes to achieve this

Same type of challenge appears when designing LNS heuristics for branch-and-price, so let's return to LNS ...

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
- Common: the auxiliary problem is formed by fixing variables

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
- Common: the auxiliary problem is formed by fixing variables

Fixing variables to 0 yield the same issues as in the 0-branch

This is where IPColGen attempts to contribute

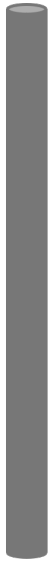
Outline of IPColGen

An LNS heuristic

- Destroy method:
  Remove columns from a current solution
- Repair method:
  - Generate columns using a special repair pricing scheme
  - Solve a repair problem = Sub-MIP

Illustrations and VRP interpretations

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

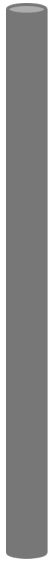


$$= \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

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

Illustrations and VRP interpretations

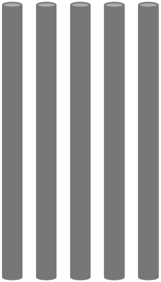
Column $=$ binary vector $(a_{ij})_{i \in I}$



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

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

Example: **feasible solution**

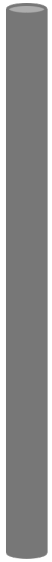


$$= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

5 routes that together visit each costumer exactly once

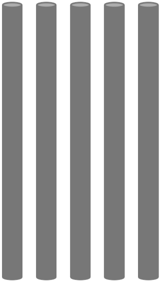
Illustrations and VRP interpretations

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

$$
\begin{aligned}
[\mathrm{MP}] \qquad \min \quad & \sum_{j\in\mathcal{J}} c_j\lambda_j, \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}} a_{ij}\lambda_j \geq 1, \quad i \in I^{\mathrm{c}}, \\
& \sum_{j\in\mathcal{J}} a_{ij}\lambda_j \leq 1, \quad i \in I^{\mathrm{p}}, \\
& (\lambda_j)_{j\in\mathcal{J}} \in \mathcal{L} \subseteq \{0,1\}^{|\mathcal{J}|},
\end{aligned}
$$

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

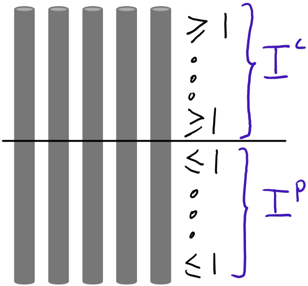
Notation

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

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

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

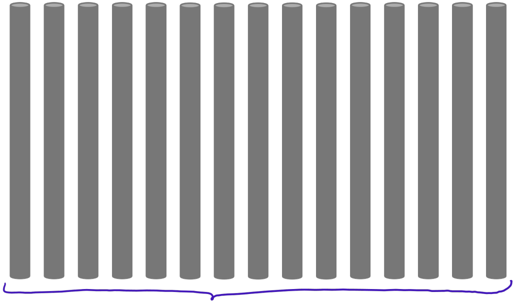
$$
(\lambda_j)_{j \in \mathcal{J}} \in \mathcal{L} \subseteq \{0,1\}^{|\mathcal{J}|},
$$



LNS – Destroy method

Columns in RMP:

$J_q, \; q \in Q$

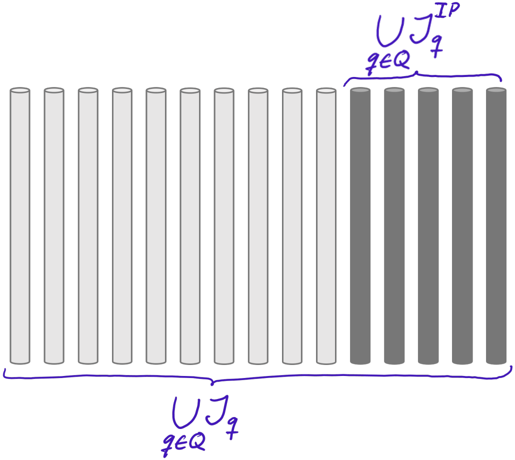


$\bigcup_{q \in Q} J_q$

LNS – Destroy method

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

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



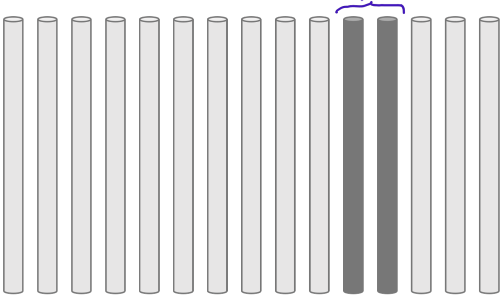
LNS – Destroy method

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

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

Destroy method = Remove active columns

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



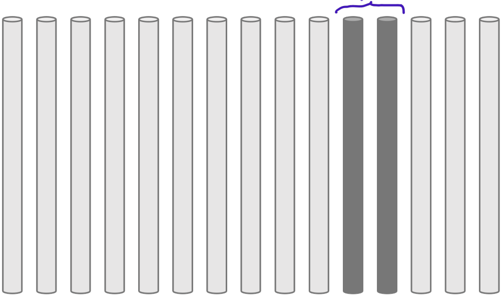
LNS – Destroy method

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

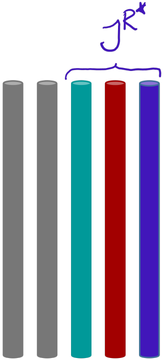
Solve [REP] over the set $J^{\mathrm{R}} = \mathcal{J}$ (all possible columns)

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

LNS – "Ideal" repair method

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

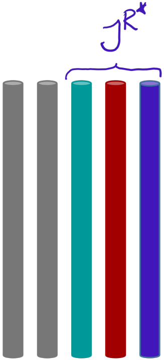
$$
\begin{aligned}
\text{[REP]} \qquad \min \quad & \sum_{j \in J^{\mathrm{R}}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in J^{\mathrm{R}}} a_{ij}\lambda_j \geq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{c}}, \\
& \sum_{j \in J^{\mathrm{R}}} a_{ij}\lambda_j \leq 1 - \sum_{j \in \hat{J}}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



LNS – "Ideal" repair method

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

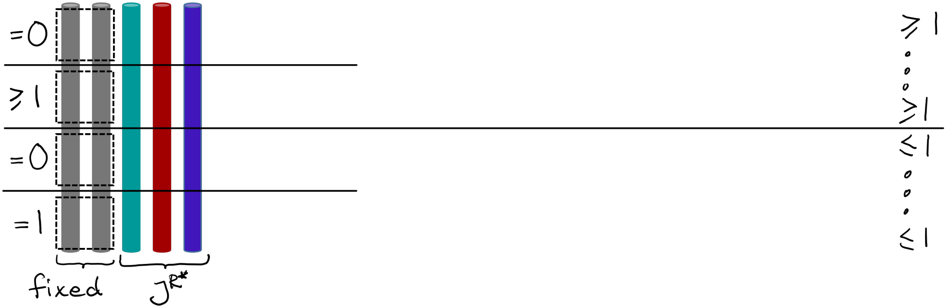
$$
\begin{aligned}
\text{[REP]} \qquad \min \quad & \sum_{j \in J^{\mathrm{R}}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in J^{\mathrm{R}}} a_{ij} \lambda_j \geq 1 - \sum_{j \in \hat{J}} a_{ij}, \quad i \in I^{\mathrm{c}}, \\
& \sum_{j \in J^{\mathrm{R}}} a_{ij} \lambda_j \leq 1 - \sum_{j \in \hat{J}}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



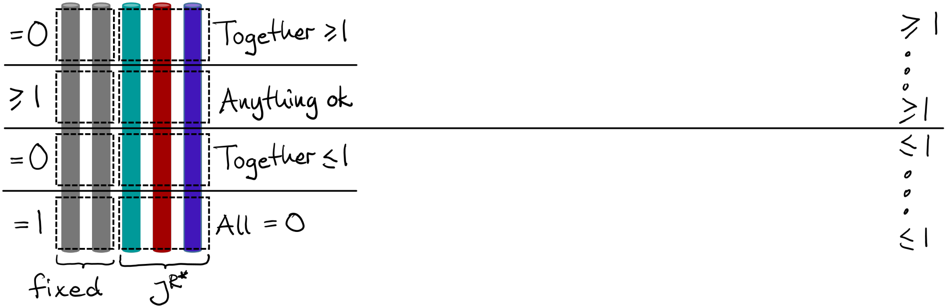
NOT reasonable in practice!

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

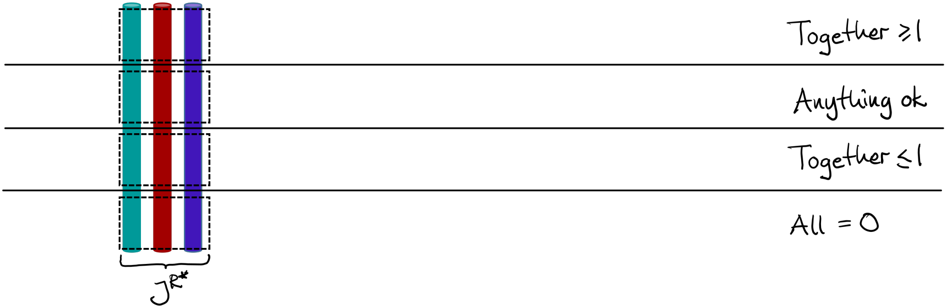
Properties of J^{R^*} and desired properties of J^R



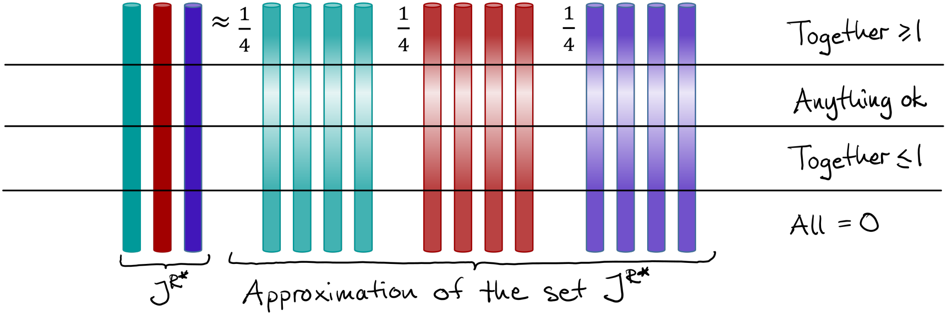
Properties of J^{R^*} and desired properties of J^R



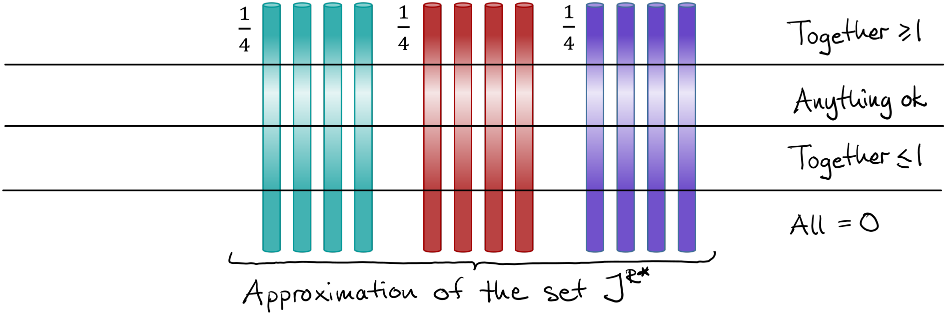
Properties of J^{R^*} and desired properties of J^R



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

Repair pricing

Pricing problem q in iteration l

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

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

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

$$
\begin{aligned}
[\text{REP-CG}_{ql}] \quad \min \quad & c - \sum_{i \in I^{\mathrm{c}}} \gamma \bar{u}_i a_i + \sum_{i \in I^{\mathrm{p}}} \gamma \bar{u}_i a_i + \\
& + \sum_{i \in \hat{I}^{\mathrm{p}1}} M a_i - \sum_{i \in \hat{I}^{\mathrm{c}0}} \beta_{il} a_i + \sum_{i \in \hat{I}^{\mathrm{p}0}} \beta_{il} a_i \\
\text{s.t.} \quad & (c, a) \in \mathcal{A}_q .
\end{aligned}
$$

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

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.

Implementation in GCG module of SCIP

IPColGen is implemented as part of the B&P&C scheme in GCG

- Apply in root node

Implementation in GCG module of SCIP

IPColGen is implemented as part of the B&P&C scheme in GCG

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

Implementation in GCG module of SCIP

IPColGen is 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)

Implementation in GCG module of SCIP

IPColGen is 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)

Implementation in GCG module of SCIP

IPColGen is 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

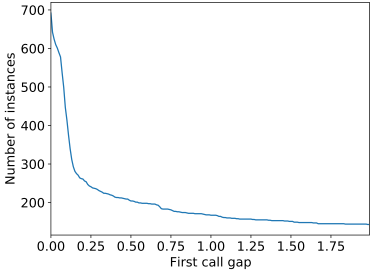
- Bin packing
- Capacitated p-median
- Generalised assignment
- Vertex coloring
- Optimal interval scheduling

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

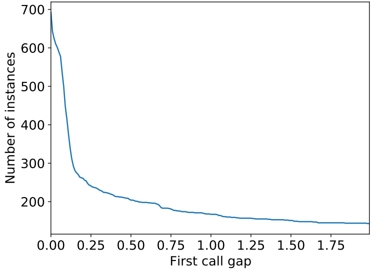


Instances with known block diagonal structures

Results for about 700 instances

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

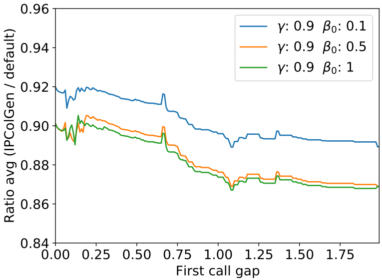
Instance characteristics



Show results for some parameter settings γ and β

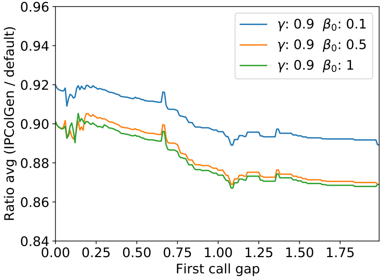
Results: Instances with known block diagonal structures

Final optimality gap

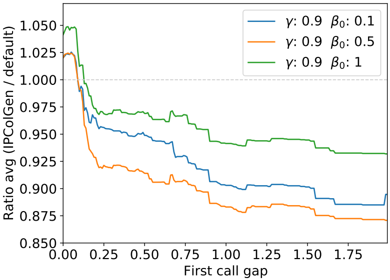


Results: Instances with known block diagonal structures

Final optimality gap

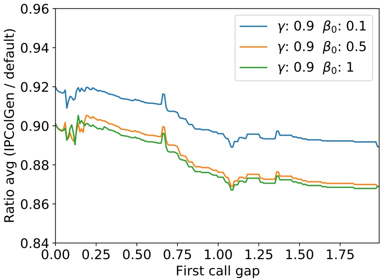


Primal integral

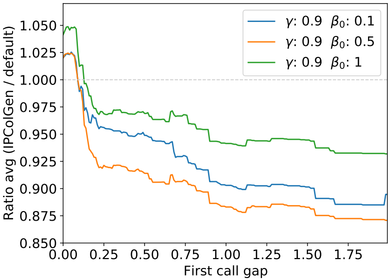


Results: Instances with known block diagonal structures

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

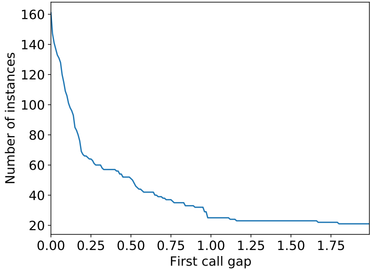
Automatic structure detection & D-W decomposition in GCG

Instances from MIPLIB 2017

Results for about 160 instances with known solution and tags

- Decomposition
- Set covering
- Set packing
- Set partitioning

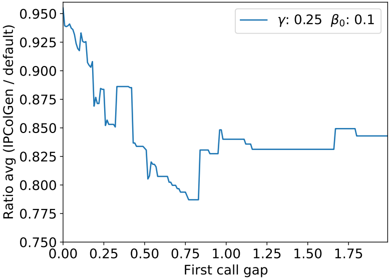
Instance characteristics



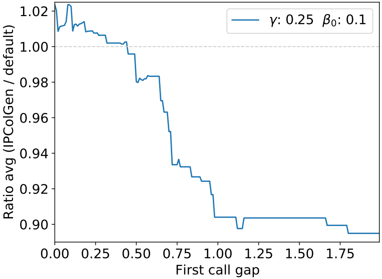
Automatic structure detection & D-W decomposition in GCG

Results: Instances from MIPLIB 2017

Final primal gap

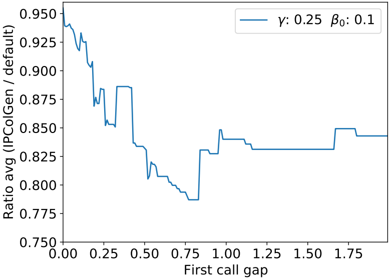


Primal integral

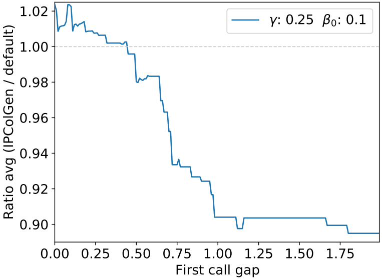


Results: Instances from MIPLIB 2017

Final primal gap



Primal integral



Same type of results as for instances with known structure!

Conclusions

IPColGen behaves as intended:
Helps GCG/SCIP finding high-quality integer solutions & improves computational performance for difficult instances

Conclusions

IPColGen behaves as intended:
Helps GCG/SCIP finding high-quality integer solutions & improves computational performance for difficult instances

Paper also includes

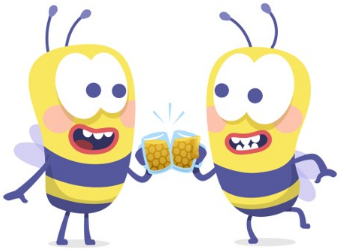
- Detailed derivation of pricing scheme
- More tests + performance measures
- Analysis for different parameter settings
- An extension of the restricted master heuristic

Room for several improvements of both theory and implementation

Final notes and acknowledgments

April 2023:
Published in Mathematical Programming Computation

Elina Rönnberg: funding from The Center for Industrial Information Technology (CENIIT)



Thanks for listening!

Open positions on the horizon

- **Assistant Professor in non-linear programming**
  - Needs to be a touch of AI, e.g. optimsiation for learning
  - Funding: 80% research in 5 years + PhD student or 2 postdocs
- **Any type of Professor in MIP/discrete optimisation**
  - Preferably someone who wants to collaborate with me =)
  - Nice if interested in combining with data-driven methods and has interest in both theory, methods and applications
  - Funding: 80% research in 5 years +
    can take part in projects/supervision in my group

Both are permanent positions

(as Assistant professor you can get kicked out after 5 years if duties are neglected)

I need to get in contact with candidates before announcing!