1st Brazilian Workshop on Interior Point Methods

27-28 April, 2015 - Campinas, Brazil

The advantages of interior point methods for decomposition techniques

Pedro Munari

Universidade Federal de São Carlos (UFSCar)

munari@dep.ufscar.br

FAPESP CNPq

Outline

- Decomposition techniques;

Outline

- Decomposition techniques;
- Column generation technique and branch-and-price methods;

Outline

- Decomposition techniques;
- Column generation technique and branch-and-price methods;
- Stabilized column generation using the interior point method;

Outline

- Decomposition techniques;
- Column generation technique and branch-and-price methods;
- Stabilized column generation using the interior point method;
- Interior point branch-price-and-cut;

Outline

- Decomposition techniques;
- Column generation technique and branch-and-price methods;
- Stabilized column generation using the interior point method;
- Interior point branch-price-and-cut;
- Computational results;

Outline

- Decomposition techniques;
- Column generation technique and branch-and-price methods;
- Stabilized column generation using the interior point method;
- Interior point branch-price-and-cut;
- Computational results;
- Conclusions.

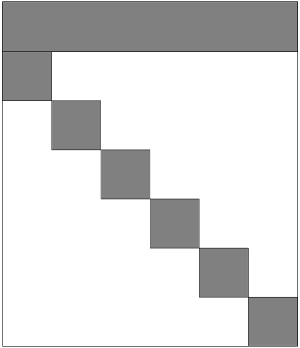
Large-scale problems

- A formulation that challenges state-of-the-art implementations;

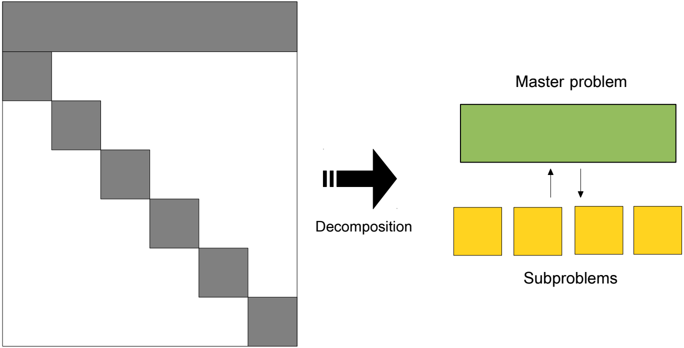
Large-scale problems

- A formulation that challenges state-of-the-art implementations;
- Special structure in the coefficient matrix, which allows a decomposition (e.g. Dantzig-Wolfe decomposition).

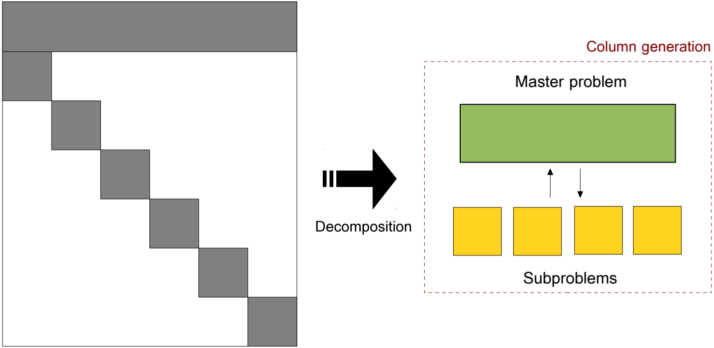
Large-scale problems



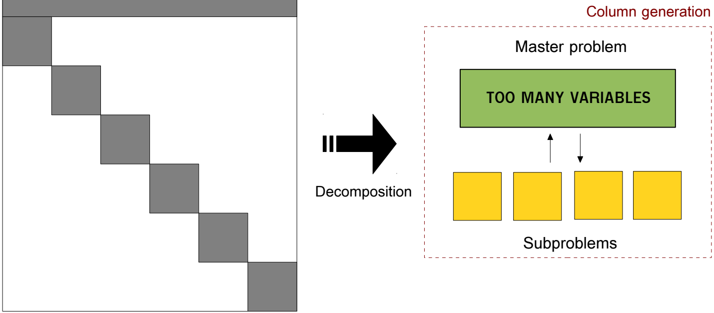
Large-scale problems



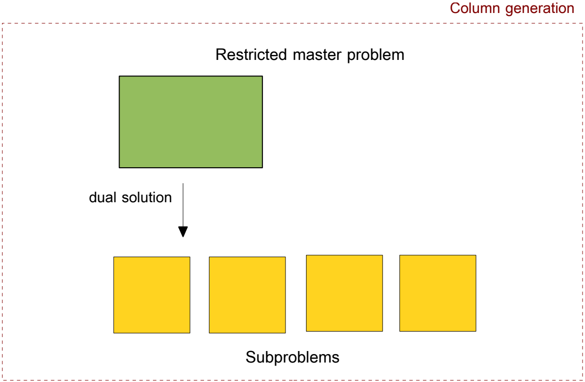
Large-scale problems



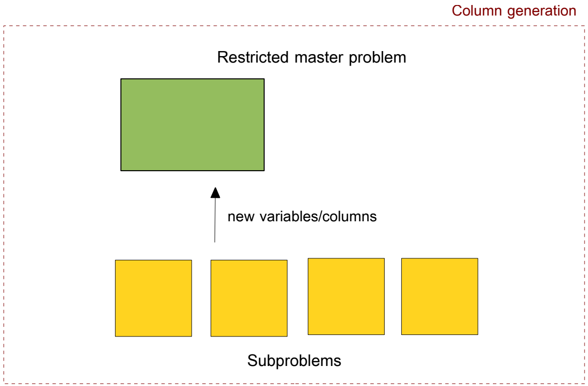
Large-scale problems



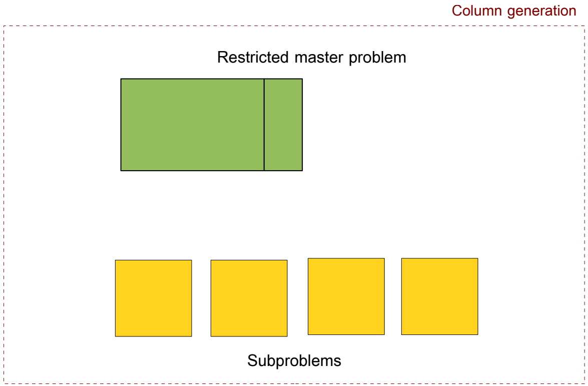
Large-scale problems



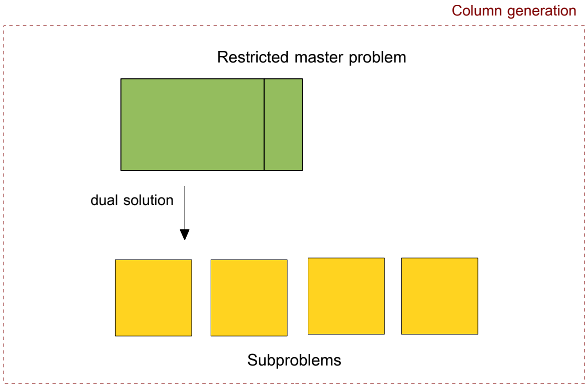
Large-scale problems



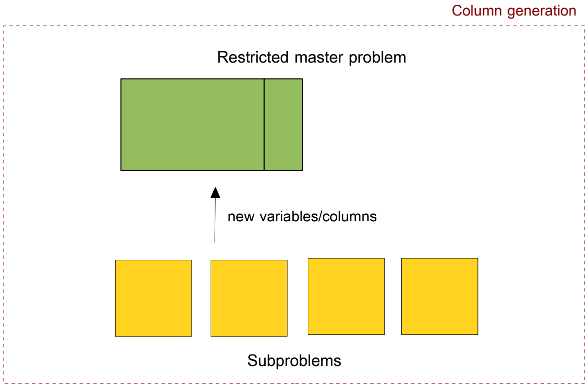
Large-scale problems



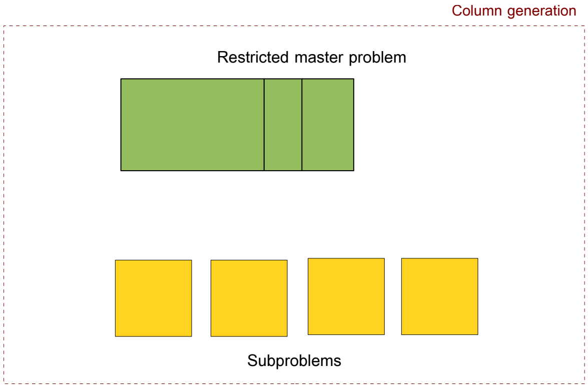
Large-scale problems



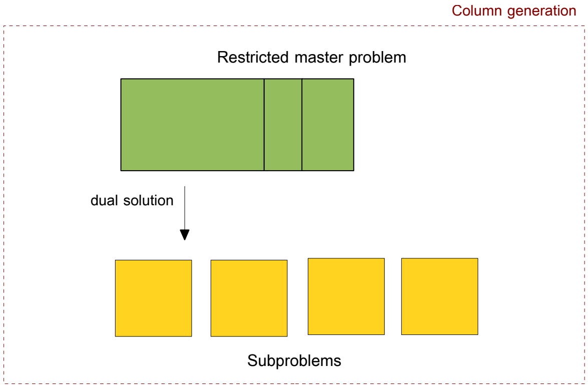
Large-scale problems



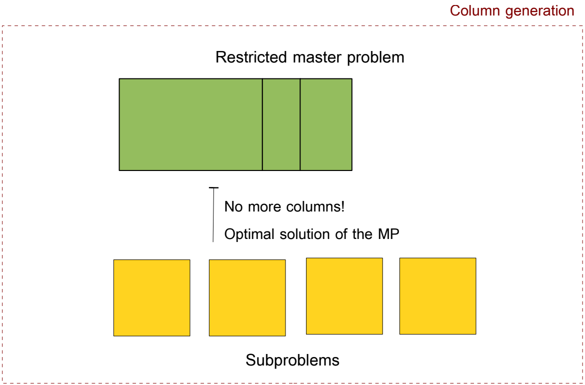
Large-scale problems



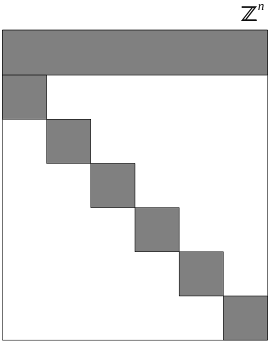
Large-scale problems



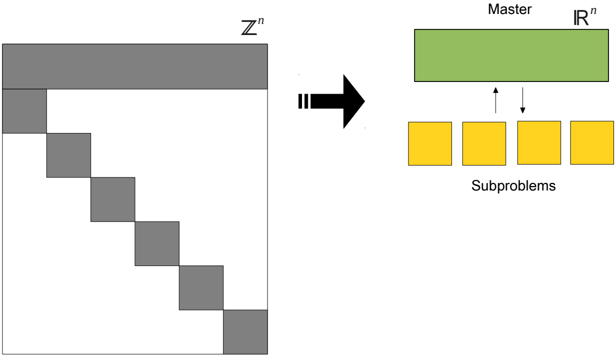
Large-scale problems



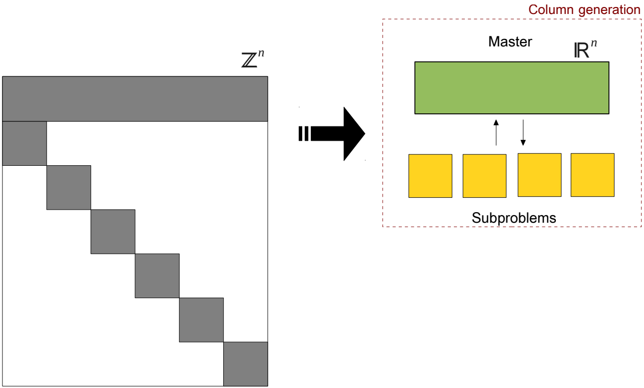
Large-scale discrete problems



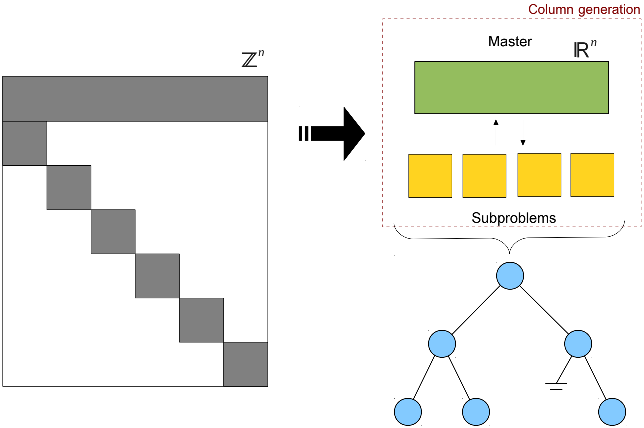
Large-scale discrete problems



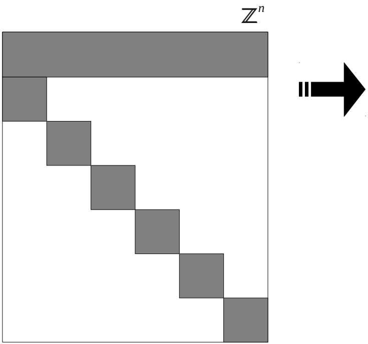
Large-scale discrete problems

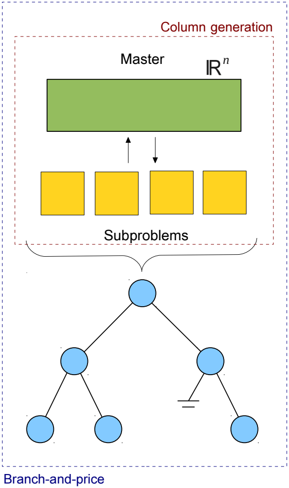


Large-scale discrete problems



Large-scale discrete problems





Large-scale problems

- A formulation that challenges state-of-the-art implementations;
- Special structure in the coefficient matrix, which allows a decomposition (e.g. Dantzig-Wolfe decomposition).

Large-scale problems

- A formulation that challenges state-of-the-art implementations;
- Special structure in the coefficient matrix, which allows a decomposition (e.g. Dantzig-Wolfe decomposition).
- In the context of column generation and branch-and-price, we have to solve hundreds of thousands of LP problems in sequence;

Large-scale problems

- A formulation that challenges state-of-the-art implementations;
- Special structure in the coefficient matrix, which allows a decomposition (e.g. Dantzig-Wolfe decomposition).
- In the context of column generation and branch-and-price, we have to solve hundreds of thousands of LP problems in sequence;
- This way, it is important to use an efficient LP algorithm;

Large-scale problems

- A formulation that challenges state-of-the-art implementations;
- Special structure in the coefficient matrix, which allows a decomposition (e.g. Dantzig-Wolfe decomposition).
- In the context of column generation and branch-and-price, we have to solve hundreds of thousands of LP problems in sequence;
- This way, it is important to use an efficient LP algorithm;
- We should exploit also additional advantages offered by the algorithm.

Column generation

- We are interested in solving a linear programming problem, called the Master Problem (MP):

$$\begin{aligned} z^\star := \min \quad & \sum_{j \in N} c_j \lambda_j, \\ \text{s.t.} \quad & \sum_{j \in N} a_j \lambda_j = b, \\ & \lambda_j \geq 0, \qquad \forall j \in N. \end{aligned}$$

- N is too big;

Column generation

- We are interested in solving a linear programming problem, called the Master Problem (MP):

$$
\begin{aligned}
z^\star := \min \quad & \sum_{j \in N} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in N} a_j \lambda_j = b, \\
& \lambda_j \geq 0, \qquad \forall j \in N.
\end{aligned}
$$

- N is too big;
- The columns (c_j, a_j) are not known explicitly;

Column generation

- We are interested in solving a linear programming problem, called the Master Problem (MP):

$$
\begin{aligned}
z^\star := \text{min} \quad & \sum_{j \in N} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in N} a_j \lambda_j = b, \\
& \lambda_j \geq 0, \qquad \forall j \in N.
\end{aligned}
$$

- N is too big;
- The columns (c_j, a_j) are not known explicitly;
- We know how to generate them!

Column generation

- The Restricted Master Problem (RMP):

$$
\begin{aligned}
z_{RMP} := \min \quad & \sum_{j \in \overline{N}} c_j \lambda_j, \\
\text{s.t.} \quad & \sum_{j \in \overline{N}} a_j \lambda_j = b, \\
& \lambda_j \geq 0, \qquad \forall j \in \overline{N}.
\end{aligned}
$$

- with $\overline{N} \subset N$.

Column generation

- The Restricted Master Problem (RMP):

$$\begin{aligned} z_{RMP} := \min \quad & \sum_{j \in \overline{N}} c_j \lambda_j, \\ \text{s.t.} \quad & \sum_{j \in \overline{N}} a_j \lambda_j = b, \\ & \lambda_j \geq 0, \qquad \forall j \in \overline{N}. \end{aligned}$$

- with $\overline{N} \subset N$.
- Let $(\bar{\lambda}, \bar{u})$ be a primal-dual optimal solution of the RMP.

Column generation

- The Restricted Master Problem (RMP):

$$
\begin{aligned}
z_{RMP} := \min \quad & \sum_{j \in \overline{N}} c_j \lambda_j, & \\
\text{s.t.} \quad & \sum_{j \in \overline{N}} a_j \lambda_j = b, & (u) \\
& \lambda_j \geq 0, & \forall j \in \overline{N}.
\end{aligned}
$$

- with $\overline{N} \subset N$.
- Let $(\bar{\lambda}, \bar{u})$ be a primal-dual optimal solution of the RMP.

Column generation

- Solution $\bar{\lambda}$ of the RMP $\Rightarrow$ solution $\hat{\lambda}$ of the MP;

Column generation

- Solution $\bar{\lambda}$ of the RMP $\Rightarrow$ solution $\hat{\lambda}$ of the MP;
- How to know if $\hat{\lambda}$ is optimal in the MP?

Column generation

- Solution $\bar{\lambda}$ of the RMP $\Rightarrow$ solution $\hat{\lambda}$ of the MP;
- How to know if $\hat{\lambda}$ is optimal in the MP?
  - Call the pricing subproblem (oracle):

$$z_{SP} := \min\{0, c_j - \overline{u}^T a_j | (c_j, a_j) \in \mathcal{A}\}.$$

  - (c_j, a_j) are the variables in the subproblem;

Column generation

- Solution $\bar{\lambda}$ of the RMP $\Rightarrow$ solution $\hat{\lambda}$ of the MP;
- How to know if $\hat{\lambda}$ is optimal in the MP?
  - Call the pricing subproblem (oracle):

$$z_{SP} := \min\{0, c_j - \overline{u}^T a_j | (c_j, a_j) \in \mathcal{A}\}.$$

  - (c_j, a_j) are the variables in the subproblem;
- If $z_{SP} < 0$, then new columns are generated;

Column generation

- Solution $\bar{\lambda}$ of the RMP $\Rightarrow$ solution $\hat{\lambda}$ of the MP;
- How to know if $\hat{\lambda}$ is optimal in the MP?
  - Call the pricing subproblem (oracle):

$$z_{SP} := \min\{0, c_j - \overline{u}^T a_j | (c_j, a_j) \in \mathcal{A}\}.$$

  - (c_j, a_j) are the variables in the subproblem;
- If $z_{SP} < 0$, then new columns are generated;
- Otherwise, an optimal solution of the MP was found!

Standard column generation

- Optimal solutions, typically obtained by the simplex method:

Standard column generation

- Optimal solutions, typically obtained by the simplex method:

  $\Rightarrow$ Extreme points of the RMP;

Standard column generation

- Optimal solutions, typically obtained by the simplex method:

  $\Rightarrow$ Extreme points of the RMP;

- They oscillate too much between consecutive iterations;

Standard column generation

- Optimal solutions, typically obtained by the simplex method:

  $\Rightarrow$ Extreme points of the RMP;

- They oscillate too much between consecutive iterations;

  $\Rightarrow$ u^{j+1} is typically far from u^j;

Standard column generation

- Optimal solutions, typically obtained by the simplex method:

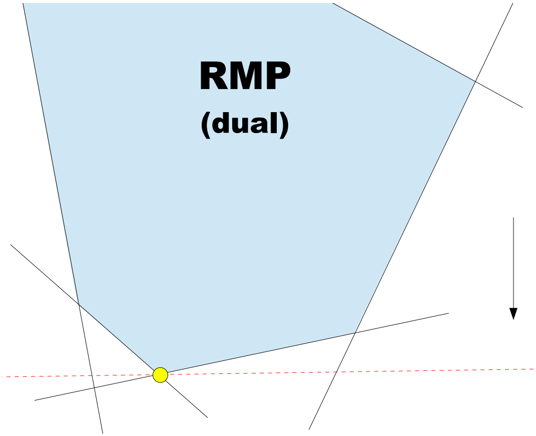
  $\Rightarrow$ Extreme points of the RMP;

- They oscillate too much between consecutive iterations;

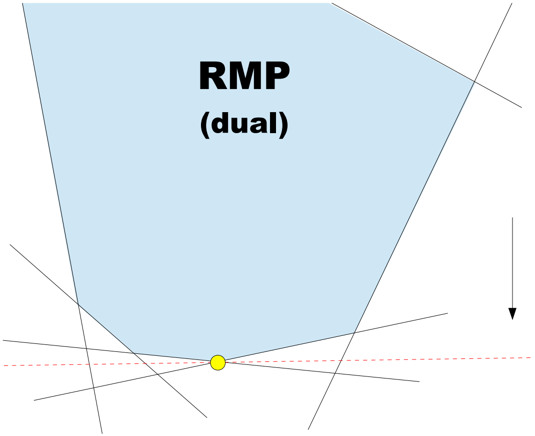
  $\Rightarrow$ u^{j+1} is typically far from u^{j};

- Slow convergence of the method.

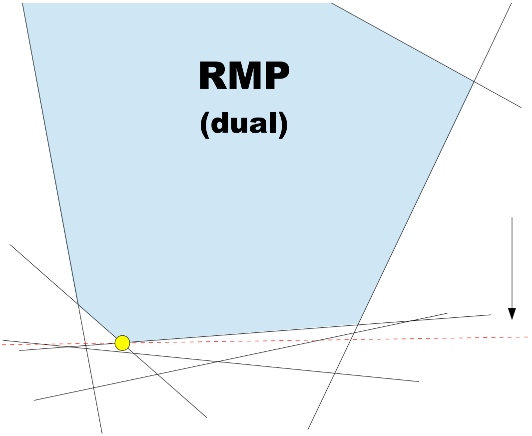
Standard column generation



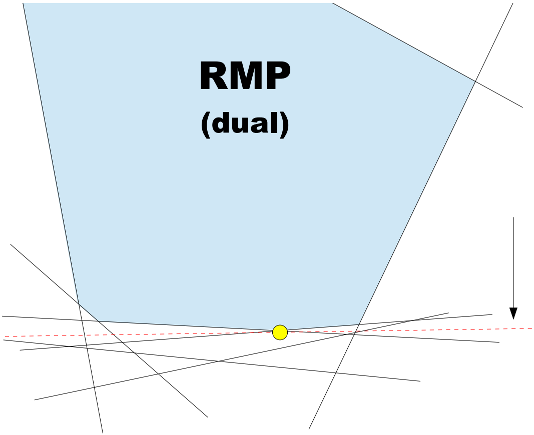
Standard column generation



Standard column generation

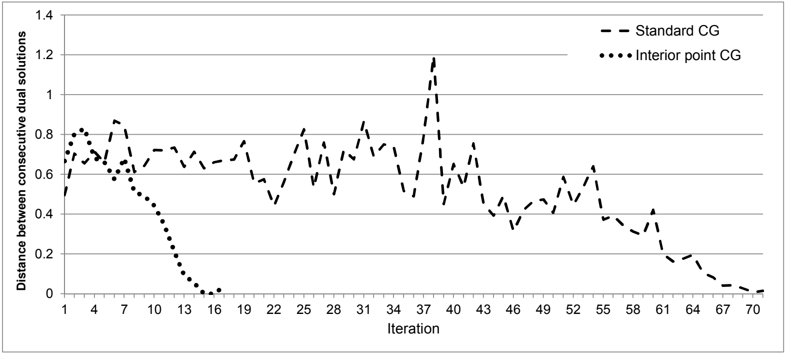


Standard column generation



Oscillation in a real instance

$\|u^j - u^{j+1}\|_2$, for each iteration j:



Column generation variants

- Stabilization techniques: avoid extreme solutions!

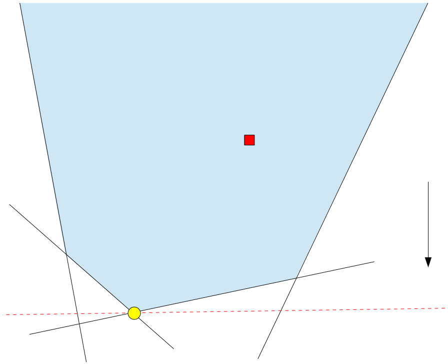
Column generation variants

- Stabilization techniques: avoid extreme solutions!

  $\Rightarrow$ use a point in the interior of the feasible set;

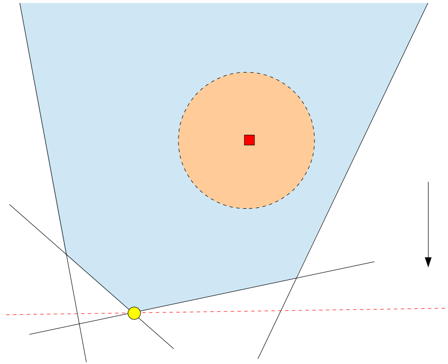
Column generation variants

- Stability center: prohibit the next dual solution to go far from it;



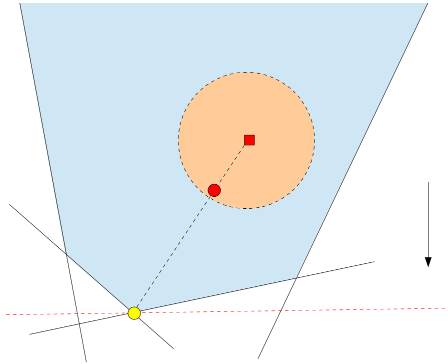
Column generation variants

- Stability center: prohibit the next dual solution to go far from it;



Column generation variants

- Stability center: prohibit the next dual solution to go far from it;



Column generation variants

- Stabilization techniques: avoid extreme solutions!

  $\Rightarrow$ use a point in the interior of the feasible set;

- Most of them: modify the master problem!

Column generation variants

- Stabilization techniques: avoid extreme solutions!

  $\Rightarrow$ use a point in the interior of the feasible set;

- Most of them: modify the master problem!

- Add variables, bounds, constraints, penalties, ...

  $\Rightarrow$ The master problem may become more difficult to solve;

Column generation variants

- Stabilization techniques: avoid extreme solutions!

  $\Rightarrow$ use a point in the interior of the feasible set;

- Most of them: modify the master problem!

- Add variables, bounds, constraints, penalties, ...

  $\Rightarrow$ The master problem may become more difficult to solve;

  $\Rightarrow$ Some of them may be difficult to implement;

Column generation variants

- Stabilization techniques: avoid extreme solutions!

  $\Rightarrow$ use a point in the interior of the feasible set;

- Most of them: modify the master problem!
- Add variables, bounds, constraints, penalties, ...

  $\Rightarrow$ The master problem may become more difficult to solve;

  $\Rightarrow$ Some of them may be difficult to implement;

  $\Rightarrow$ Several parameters to tune.

Column generation variants

- The column generation is more efficient when based on well-centered interior points of the feasible set;

Column generation variants

- The column generation is more efficient when based on well-centered interior points of the feasible set;
- So, why not using an interior point method?

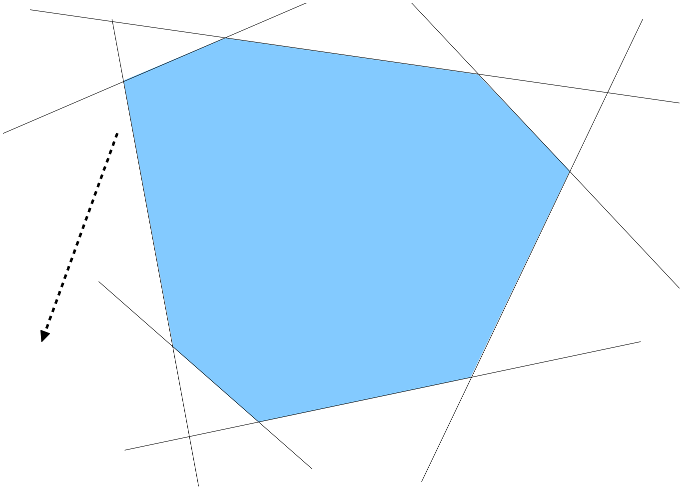
Column generation variants

- The column generation is more efficient when based on well-centered interior points of the feasible set;
- So, why not using an interior point method?
- This is straightforward: does not require any changes in the RMP nor parameter adjustments;

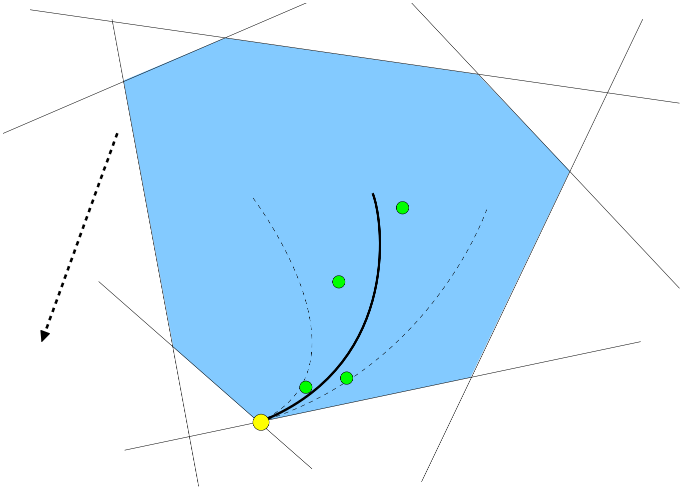
Column generation variants

- The column generation is more efficient when based on well-centered interior points of the feasible set;
- So, why not using an interior point method?
- This is straightforward: does not require any changes in the RMP nor parameter adjustments;
- Interior point methods will provided naturally stable solutions.

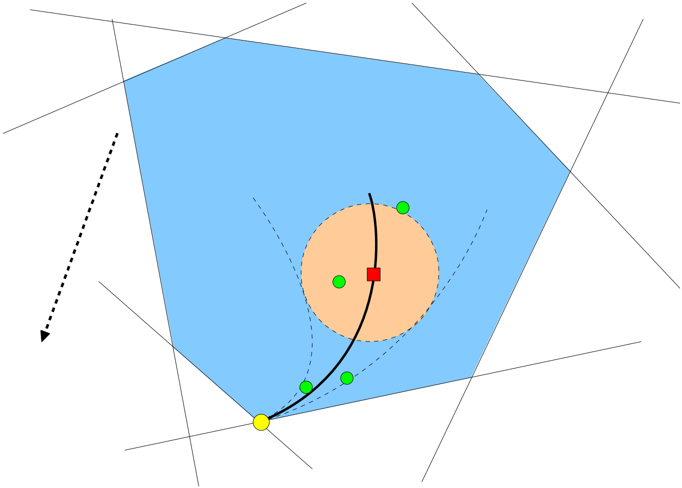
Interior point method



Interior point method



Interior point method



Non-optimal solutions from interior point method

Advantages:

- We save time, as we stop early;

Non-optimal solutions from interior point method

Advantages:

- We save time, as we stop early;
- The solution is well-centered in the feasible set;

Non-optimal solutions from interior point method

Advantages:

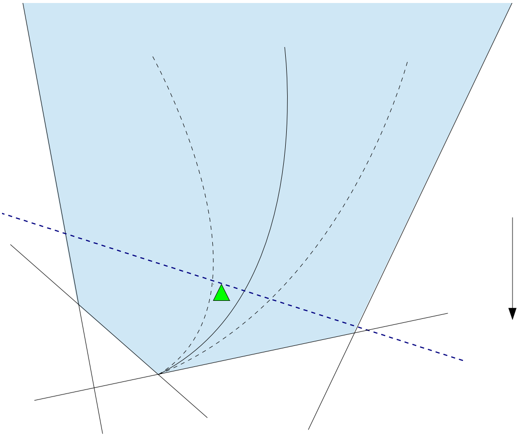
- We save time, as we stop early;
- The solution is well-centered in the feasible set;
- Early termination: good sub-optimal solutions.

Non-optimal solutions from interior point method

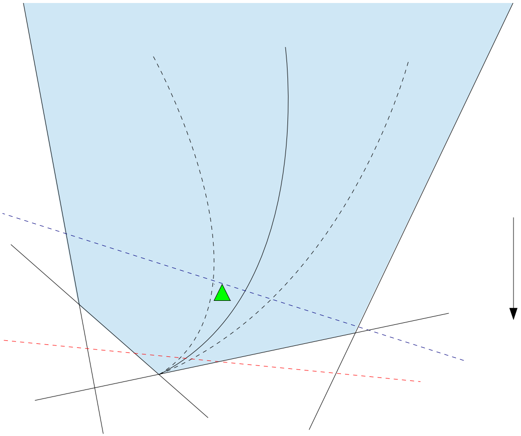
Advantages:

- We save time, as we stop early;
- The solution is well-centered in the feasible set;
- Early termination: good sub-optimal solutions.
- The column corresponds to a deeper cut in the dual space.

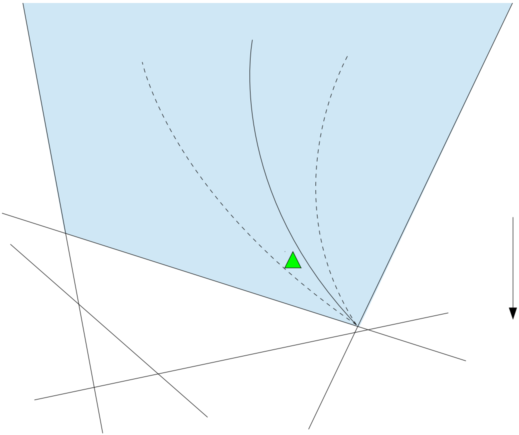
Non-optimal solutions from interior point method



Non-optimal solutions from interior point method



Non-optimal solutions from interior point method



Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;

Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;
- Suboptimal solution $(\tilde{\lambda}, \tilde{u})$ (ε-optimal solution): we stop the interior point method with optimality tolerance ε.

Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;
- Suboptimal solution $(\tilde{\lambda}, \tilde{u})$ (ε-optimal solution): we stop the interior point method with optimality tolerance ε.
- The distance to optimality ε is dynamically adjusted according to the relative gap;

Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;
- Suboptimal solution $(\tilde{\lambda}, \tilde{u})$ (ε-optimal solution): we stop the interior point method with optimality tolerance ε.
- The distance to optimality ε is dynamically adjusted according to the relative gap;

$$\varepsilon = \min\{\varepsilon_{\max},\ \mathsf{gap}/D\}$$

Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;
- Suboptimal solution $(\tilde{\lambda}, \tilde{u})$ (ε-optimal solution): we stop the interior point method with optimality tolerance ε.
- The distance to optimality ε is dynamically adjusted according to the relative gap;

$$\varepsilon = \min\{\varepsilon_{\max},\ \mathsf{gap}/D\}$$

- $\mathsf{gap} = (\mathsf{UB} - \mathsf{LB})/(1 + |\mathsf{UB}|)$;

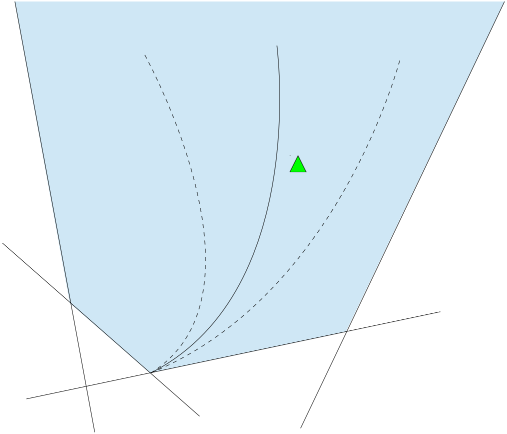
Primal-dual column generation method (PDCGM)

- Primal-dual interior point method to get primal-dual solutions;
- Suboptimal solution $(\tilde{\lambda}, \tilde{u})$ (ε-optimal solution): we stop the interior point method with optimality tolerance ε.
- The distance to optimality ε is dynamically adjusted according to the relative gap;

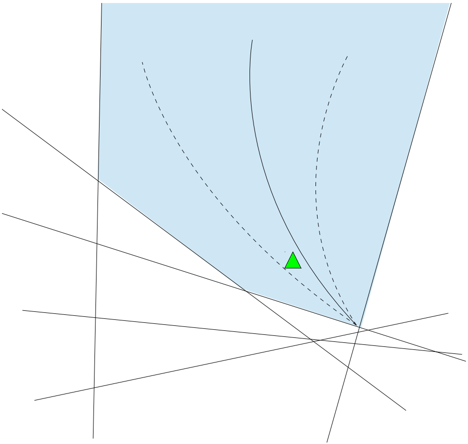
$$\varepsilon = \min\{\varepsilon_{\max},\ \mathsf{gap}/D\}$$

- $\mathsf{gap} = (\mathsf{UB} - \mathsf{LB})/(1 + |\mathsf{UB}|)$;
- D: degree of optimality (fixed);

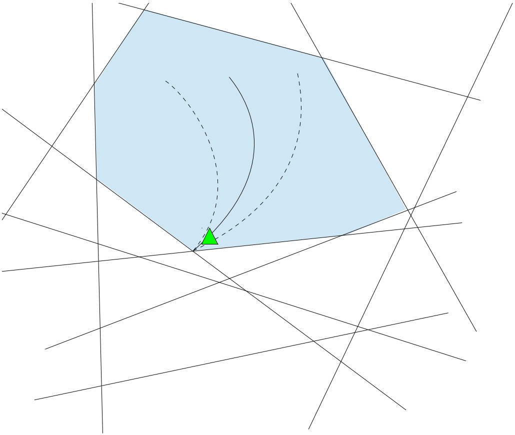
Primal-dual column generation method (PDCGM)



Primal-dual column generation method (PDCGM)



Primal-dual column generation method (PDCGM)



PDCGM: Algorithm

PDCGM: Algorithm

1. **Input:** Initial RMP; parameters κ, $\varepsilon_{\max}$, $D > 1$, $\delta > 0$, .
2. set LB $= -\infty$, UB $= \infty$, gap $= \infty$, $\varepsilon = 0.5$;
3. while (gap $> \delta$) do
4. find a well-centered ε-optimal solution $(\tilde{\lambda}, \tilde{u})$ of the RMP;
5. UB $= \min\{\text{UB}, \tilde{z}_{RMP}\}$;
6. call the oracle with the query point $\tilde{u}$;
7. LB $= \max\{\text{LB}, \kappa \tilde{z}_{SP} + b^T \tilde{u}\}$;
8. gap $= (\text{UB} - \text{LB})/(1 + |\text{UB}|)$;
9. $\varepsilon = \min\{\varepsilon_{\max},\ \text{gap}/D\}$;
10. if $(\tilde{z}_{SP} < 0)$ then add the new columns into the RMP;
11. end(while)

PDCGM: Algorithm

1. **Input:** Initial RMP; parameters κ, $\varepsilon_{\max}$, $D > 1$, $\delta > 0$, .
2. set LB $= -\infty$, UB $= \infty$, gap $= \infty$, $\varepsilon = 0.5$;
3. while (gap $> \delta$) do
4. find a well-centered ε-optimal solution $(\tilde{\lambda}, \tilde{u})$ of the RMP;
5. UB $= \min\{\text{UB}, \tilde{z}_{RMP}\}$;
6. call the oracle with the query point $\tilde{u}$;
7. LB $= \max\{\text{LB}, \kappa \tilde{z}_{SP} + b^T \tilde{u}\}$;
8. gap $= (\text{UB} - \text{LB})/(1 + |\text{UB}|)$;
9. $\varepsilon = \min\{\varepsilon_{\max}, \text{ gap}/D\}$;
10. if $(\tilde{z}_{SP} < 0)$ then add the new columns into the RMP;
11. end(while)

PDCGM: Algorithm

- Implementation in C, using interior point solver HOPDM;

PDCGM: Algorithm

- Implementation in C, using interior point solver HOPDM;
- Publicly available code

  `http://www.maths.ed.ac.uk/~gondzio/software/pdcgm.html`

PDCGM: Algorithm

- Implementation in C, using interior point solver HOPDM;
- Publicly available code

  `http://www.maths.ed.ac.uk/~gondzio/software/pdcgm.html`
- Source-code examples are provided for 6 different applications:
  - Cutting stock problem;
  - Vehicle routing problem;
  - Capacitated lot sizing;
  - Multiple kernel learning;
  - Two-stage stochastic programming;
  - Multicommodity network flow.

European Journal of Operational Research 224 (2013) 41–51

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

New developments in the primal–dual column generation technique

Jacek Gondzio [a], Pablo González-Brevis [a,*,1], Pedro Munari [b,2]

[a] *School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom*
[b] *Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, Cx. Postal 668, CEP 13560-970 São Carlos, SP, Brazil*

- Theoretical and computational results;

European Journal of Operational Research 224 (2013) 41–51

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

ELSEVIER

Discrete Optimization

New developments in the primal–dual column generation technique

Jacek Gondzio [a], Pablo González-Brevis [a,*,1], Pedro Munari [b,2]

[a] School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
[b] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, Cx. Postal 668, CEP 13560-970 São Carlos, SP, Brazil

- Theoretical and computational results;
- Linear relaxations of combinatorial optimization problems:

European Journal of Operational Research 224 (2013) 41–51

Contents lists available at SciVerse ScienceDirect

ELSEVIER

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

New developments in the primal–dual column generation technique

Jacek Gondzio [a], Pablo González-Brevis [a,*,1], Pedro Munari [b,2]

[a] School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
[b] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, Cx. Postal 668, CEP 13560-970 São Carlos, SP, Brazil

- Theoretical and computational results;
- Linear relaxations of combinatorial optimization problems:
    - Cutting stock problem (CSP);

European Journal of Operational Research 224 (2013) 41–51

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

New developments in the primal–dual column generation technique

Jacek Gondzio [a], Pablo González-Brevis [a,*,1], Pedro Munari [b,2]

[a] School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
[b] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, Cx. Postal 668, CEP 13560-970 São Carlos, SP, Brazil

- Theoretical and computational results;
- Linear relaxations of combinatorial optimization problems:
  - Cutting stock problem (CSP);
  - Vehicle routing problem with time windows (VRPTW);

European Journal of Operational Research 224 (2013) 41–51

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

ELSEVIER

Discrete Optimization

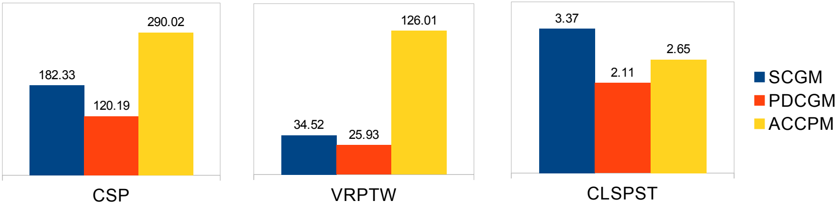
New developments in the primal–dual column generation technique

Jacek Gondzio [a], Pablo González-Brevis [a,*,1], Pedro Munari [b,2]

[a] School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
[b] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, Cx. Postal 668, CEP 13560-970 São Carlos, SP, Brazil

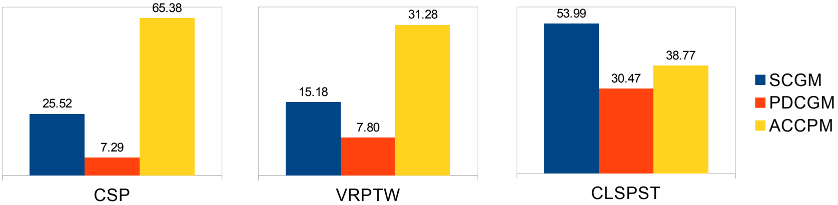
- Theoretical and computational results;
- Linear relaxations of combinatorial optimization problems:
  - Cutting stock problem (CSP);
  - Vehicle routing problem with time windows (VRPTW);
  - Capacitated lot sizing with setup times (CLSPST).

Number of iterations



Relative to PDCGM	CSP	VRPTW	CLSPST
SCGM	1.52	1.33	1.60
ACCPM	2.41	4.86	1.26

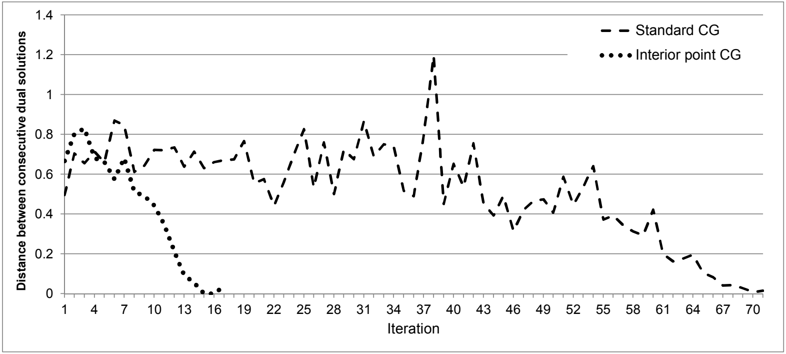
CPU time (s)



Relative to PDCGM	CSP	VRPTW	CLSPST
SCGM	3.50	1.95	1.26
ACCPM	8.97	4.01	1.27

Oscillation in a VRPTW instance (Solomon C207)

$\|u^j - u^{j+1}\|_2$, for each iteration j:



Convex optimization problems

Convex optimization problems

- The bottleneck in the previously addressed problems was in the subproblems: combinatorial optimization problems;

Convex optimization problems

- The bottleneck in the previously addressed problems was in the subproblems: combinatorial optimization problems;
- Would PDCGM be successful in problems with easy subproblems?

Convex optimization problems

- The bottleneck in the previously addressed problems was in the subproblems: combinatorial optimization problems;
- Would PDCGM be successful in problems with easy subproblems?
- We have selected three classes of problems:
  - Two-stage stochastic programming (TSSP);
  - Multiple kernel learning (MKL);
  - Multicommodity network flow (MCNF).

Convex optimization problems

- The bottleneck in the previously addressed problems was in the subproblems: combinatorial optimization problems;
- Would PDCGM be successful in problems with easy subproblems?
- We have selected three classes of problems:
  - Two-stage stochastic programming (TSSP);
  - Multiple kernel learning (MKL);
  - Multicommodity network flow (MCNF).
- Gondzio, González-Brevis and Munari, *Large-scale optimization with the primal-dual column generation method*, 2014. (MPC, Under review)

Convex optimization problems

- The bottleneck in the previously addressed problems was in the subproblems: combinatorial optimization problems;
- Would PDCGM be successful in problems with easy subproblems?
- We have selected three classes of problems:
  - Two-stage stochastic programming (TSSP);
  - Multiple kernel learning (MKL);
  - Multicommodity network flow (MCNF).
- Gondzio, González-Brevis and Munari, *Large-scale optimization with the primal-dual column generation method*, 2014. (MPC, Under review)
- `http://www.maths.ed.ac.uk/~gondzio/software/pdcgm.html`

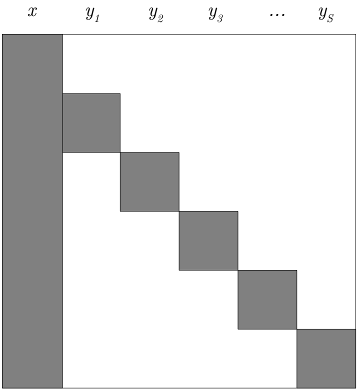
Two-stage stochastic programming problem (TSSP)

Two-stage stochastic programming problem (TSSP)

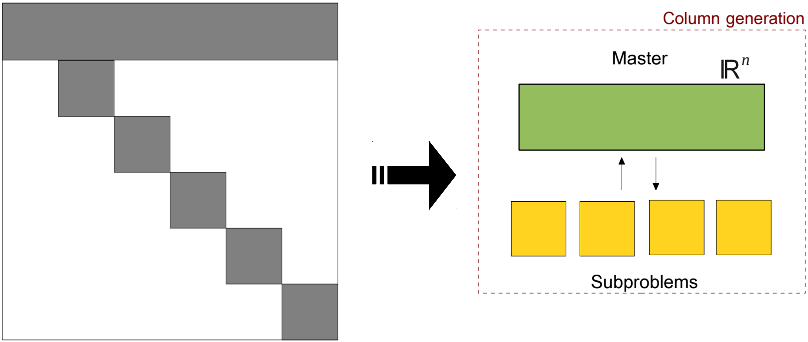
- Model many real-life situations taking uncertainty into account;
- Two interconnected problems: first-stage and recourse;
- Deterministic equivalent problem (DEP):

$$
\begin{aligned}
\min_{x,y} \quad & c^T x + \sum_{i \in \mathcal{S}} p_i q_i^T y_i, & \\
\text{s.t.} \quad & Ax = b, & \\
& T_i x + W_i y_i = h_i, & \forall i \in \mathcal{S}, \\
& x \geq 0, & \\
& y_i \geq 0, & \forall i \in \mathcal{S}.
\end{aligned}
$$

Special structure of the model



Special structure of the model



TSSP: Computational experiments

- TSSP instances that have been widely used in literature (Ariyawansa and Felt, 2004; Holmes, 1995);
- 75 instances, up to 37500 scenarios;

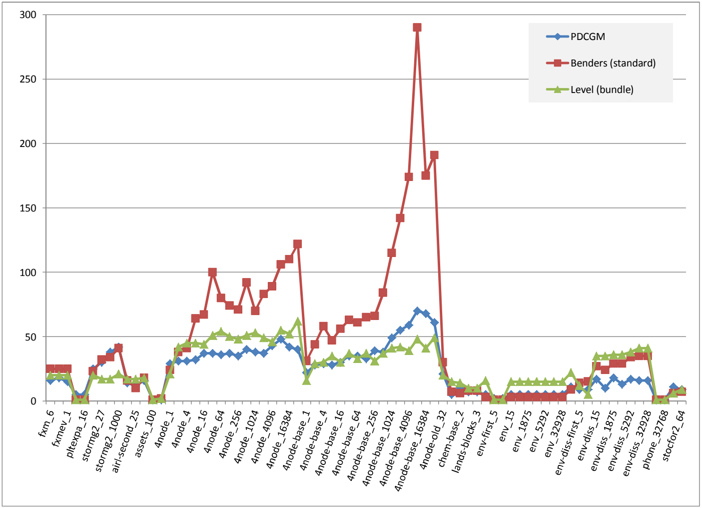
TSSP: Computational experiments

- TSSP instances that have been widely used in literature (Ariyawansa and Felt, 2004; Holmes, 1995);
- 75 instances, up to 37500 scenarios;
- PDCGM: Linux PC, Intel Core i7 2.8 GHz CPU, 8.0 GB of memory;
- We compare PDCGM with the following methods:
  - Standard cutting plane method (Benders);
  - Level-set method (bundle).

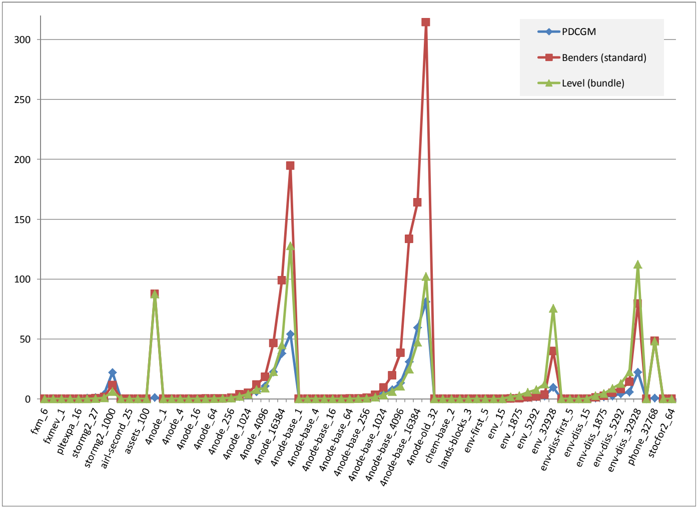
TSSP: Computational experiments

- TSSP instances that have been widely used in literature (Ariyawansa and Felt, 2004; Holmes, 1995);
- 75 instances, up to 37500 scenarios;
- PDCGM: Linux PC, Intel Core i7 2.8 GHz CPU, 8.0 GB of memory;
- We compare PDCGM with the following methods:
  - Standard cutting plane method (Benders);
  - Level-set method (bundle).
- We have taken the results of both methods from Zverovich et al. (2012), which used a Core i5 2.4 GHz CPU and 6 GB of memory.

TSSP: Number of iterations



TSSP: CPU Time (seconds)



Computers & Operations Research 40 (2013) 2026–2036

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Using the primal-dual interior point algorithm within the branch-price-and-cut method

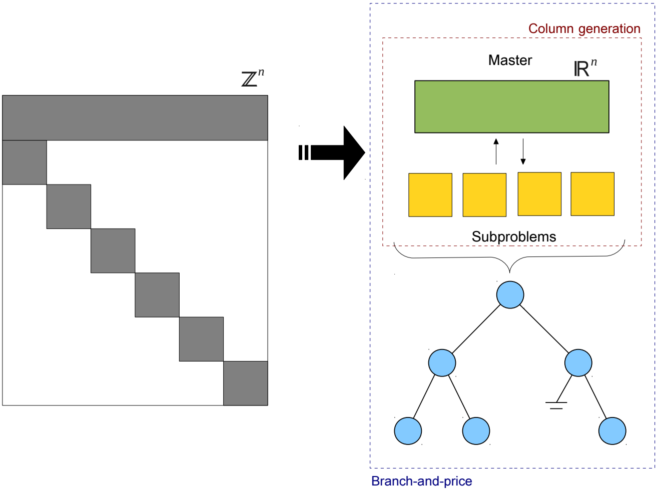
Pedro Munari [a,*,1], Jacek Gondzio [b]

[a] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Av. Trabalhador, São-carlense, 400 - Centro, Cx. Postal 668, CEP 13560-970, São Carlos-SP, Brazil

Interior point branch-price-and-cut method:

- Primal-dual interior point method;
- Well-centered dual solutions to generate columns and valid inequalities;
- Early termination;
- Vehicle routing problem with time windows (VRPTW);

Large-scale discrete problems



Interior point branch-price-and-cut (IPBPC)

- Vast majority of branch-price-and-cut methods are based on optimal solutions obtained with the simplex method;

Interior point branch-price-and-cut (IPBPC)

- Vast majority of branch-price-and-cut methods are based on optimal solutions obtained with the simplex method;
- Change of strategy!

Interior point branch-price-and-cut (IPBPC)

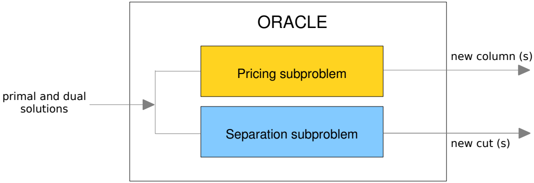
- Vast majority of branch-price-and-cut methods are based on optimal solutions obtained with the simplex method;
- Change of strategy!
- It is not just replacing a simplex-type method.

Interior point branch-price-and-cut (IPBPC)

- Vast majority of branch-price-and-cut methods are based on optimal solutions obtained with the simplex method;
- Change of strategy!
- It is not just replacing a simplex-type method.
- Rethink every piece of a standard BPC: column generation, valid inequalities, branching, ...

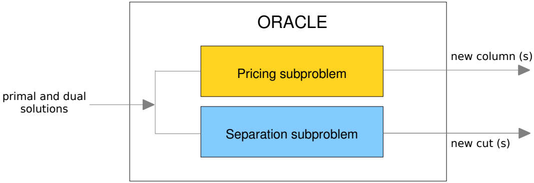
Interior point branch-price-and-cut (IPBPC)

- Primal-dual column and cut generation:
  - We modify the Oracle: two types of subproblems;



Interior point branch-price-and-cut (IPBPC)

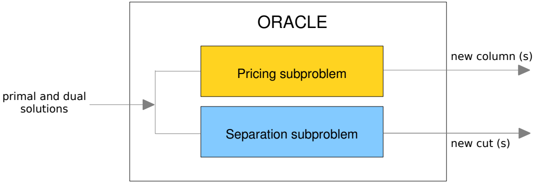
- Primal-dual column and cut generation:
  - We modify the Oracle: two types of subproblems;



- Early branching:
  - Stop CG with a loose tolerance (e.g. 10^{-3}) and branch!

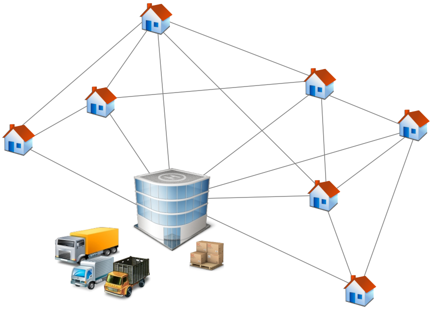
Interior point branch-price-and-cut (IPBPC)

- Primal-dual column and cut generation:
  - We modify the Oracle: two types of subproblems;

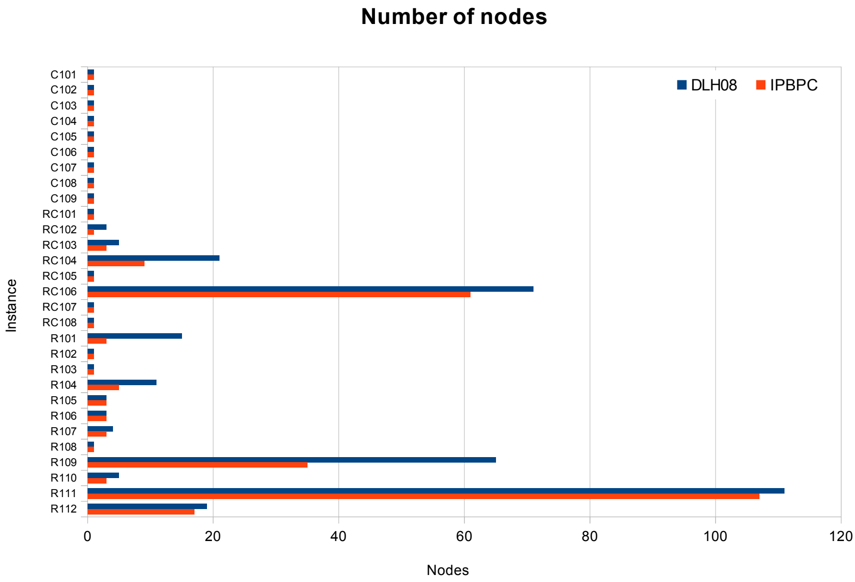


- Early branching:
  - Stop CG with a loose tolerance (e.g. 10^{-3}) and branch!
- In all cases: we use a well-centered suboptimal solution.

Vehicle Routing Problem with Time Windows (VRPTW)



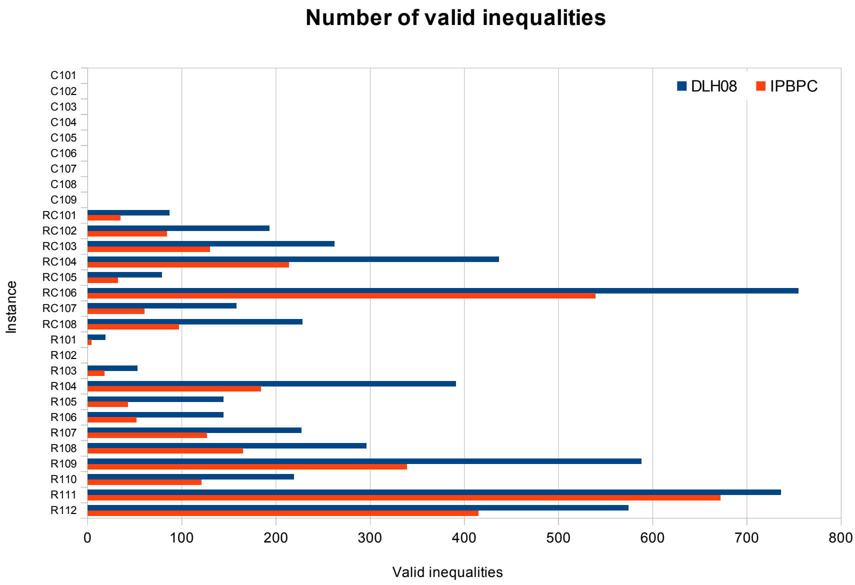
- Interior point branch-price-and-cut (IPBPC);
- The IPBPC performance was compared to the best results that were available in the literature for a *simplex-based* BPC:
  - DLH08: *Desaulniers, Lessard and Hadjar (2008), Transp. Science.*


Number of nodes
DLH08
IPBPC
Instance
C101
C102
C103
C104
C105
C106
C107
C108
C109
RC101
RC102
RC103
RC104
RC105
RC106
RC107
RC108
R101
R102
R103
R104
R105
R106
R107
R108
R109
R110
R111
R112
0
20
40
60
80
100
120
Nodes

Comparing to a simplex-based BPC

Number of nodes

	DLH08	**IPBPC**	**Ratio**
C1	9	9	1.00
RC1	104	78	1.33
R1	239	182	1.31
	352	269	1.31

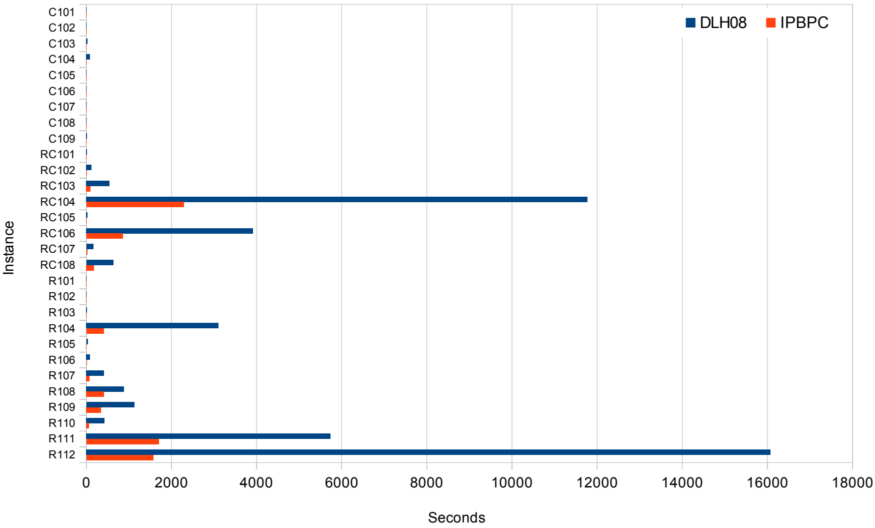

Number of valid inequalities
DLH08
IPBPC
Instance
C101
C102
C103
C104
C105
C106
C107
C108
C109
RC101
RC102
RC103
RC104
RC105
RC106
RC107
RC108
R101
R102
R103
R104
R105
R106
R107
R108
R109
R110
R111
R112
0
100
200
300
400
500
600
700
800
Valid inequalities

Comparing to a simplex-based BPC

Number of valid inequalities

	DLH08	IPBPC	Ratio
C1	0	0	1.00
RC1	2199	1191	1.85
R1	3391	2140	1.58
	5590	3331	1.68

CPU time


DLH08
IPBPC
Instance
C101
C102
C103
C104
C105
C106
C107
C108
C109
RC101
RC102
RC103
RC104
RC105
RC106
RC107
RC108
R101
R102
R103
R104
R105
R106
R107
R108
R109
R110
R111
R112
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Seconds

Comparing to a simplex-based BPC

CPU time (sec)

	DLH08	**IPBPC**	**Ratio**
C1	158	28	5.69
RC1	17198	3472	4.95
R1	27928	4621	6.04
	45284	8121	5.58

Conclusions

- The primal-dual interior point method offers a natural way of stabilizing column generation;

Conclusions

- The primal-dual interior point method offers a natural way of stabilizing column generation;
- Does not require changes to the RMP nor parameter tunning;

Conclusions

- The primal-dual interior point method offers a natural way of stabilizing column generation;
- Does not require changes to the RMP nor parameter tunning;
- Computational experiments indicate that this approach is successful in different types applications;

Conclusions

- The primal-dual interior point method offers a natural way of stabilizing column generation;
- Does not require changes to the RMP nor parameter tunning;
- Computational experiments indicate that this approach is successful in different types applications;
- Reductions in the number of iterations and CPU time when compared to standard column generation, ACCPM, bundle methods.

Thank you!

Questions?

- Acknowledgments

FAPESP CNPq

- PDCGM webpage:

  `http://www.maths.ed.ac.uk/~gondzio/software/pdcgm.html`

- munari@dep.ufscar.br