The sharpest bin-packing columns

Stefano Coniglio, Fabio Furini, Fabio D'Andreagiovanni

Column Generation 2023

Outline

References

[1] S. Coniglio, F. F. and F. D'Andreagiovanni.
A lexicographic pricer for the fractional bin packing problem.
Oper. Res. Lett, 46(6), 622-628, 2019.

[2] S. Coniglio, D. Catanzaro and F. F.
On the exact separation of cover inequalities of maximum-depth.
Optim. Lett, 16(2), 622-628, 2022.

[3] R. Baldacci, S. Coniglio, J.F. Cordeau and F. F.
A Numerically-Exact Algorithm for the Bin Packing Problem.
INFORMS Journal on computing, second revision.

References

[1] S. Coniglio, F. F. and F. D'Andreagiovanni.
A lexicographic pricer for the fractional bin packing problem.
Oper. Res. Lett, 46(6), 622-628, 2019.

[2] S. Coniglio, D. Catanzaro and F. F.
On the exact separation of cover inequalities of maximum-depth.
Optim. Lett, 16(2), 622-628, 2022.

[3] R. Baldacci, S. Coniglio, J.F. Cordeau and F. F.
A Numerically-Exact Algorithm for the Bin Packing Problem.
INFORMS Journal on computing, second revision.

References

[1] S. Coniglio, F. F. and F. D'Andreagiovanni.
A lexicographic pricer for the fractional bin packing problem.
Oper. Res. Lett, 46(6), 622-628, 2019.

[2] S. Coniglio, D. Catanzaro and F. F.
On the exact separation of cover inequalities of maximum-depth.
Optim. Lett, 16(2), 622-628, 2022.

[3] R. Baldacci, S. Coniglio, J.F. Cordeau and F. F.
A Numerically-Exact Algorithm for the Bin Packing Problem.
INFORMS Journal on computing, second revision.

Fractional bin packing problem

The bin packing problem (BPP)

Given:

- a set $N = \{1, \ldots, n\}$ of n items with positive integer **weights** $w_1, \ldots, w_n$
- an unlimited number of bins with a positive integer **capacity** C

the **bin packing problem** (BPP) asks to compute the minimum number of bins that are necessary to pack all the items.

The bin packing problem (BPP)

Given:

- a set $N = \{1, \ldots, n\}$ of n items with positive integer **weights** $w_1, \ldots, w_n$
- an unlimited number of bins with a positive integer **capacity** C

the **bin packing problem** (BPP) asks to compute the minimum number of bins that are necessary to pack all the items.

A set-covering model for the BPP

A feasible (cutting) **pattern** is any subset of items respecting the bin capacity:

$$S \subseteq N \text{ with } \sum_{j \in S} w_j \leq C$$

Let $\mathscr{S}$ be the **collection of all feasible cutting patterns**:

$$\mathscr{S} = \left\{ S \subseteq N : \sum_{j \in S} w_j \leq C \right\} \text{ and } \mathscr{S}(j) = \{S \in \mathscr{S} : j \in S\}$$

The set-covering model for the BPP reads as follows:

$$\min_{y \in \{0,1\}^{|\mathscr{S}|}} \left\{ \sum_{S \in \mathscr{S}} y_S : \sum_{S \in \mathscr{S}(j)} y_S \geq 1, \ \forall j \in N \right\}$$

A very strong ILP model with an **exponential** number of binary variables!

A set-covering model for the BPP

A feasible (cutting) **pattern** is any subset of items respecting the bin capacity:

$$S \subseteq N \text{ with } \sum_{j \in S} w_j \leq C$$

Let $\mathscr{S}$ be the **collection of all feasible cutting patterns**:

$$\mathscr{S} = \left\{ S \subseteq N : \sum_{j \in S} w_j \leq C \right\} \text{ and } \mathscr{S}(j) = \{S \in \mathscr{S} : j \in S\}$$

The set-covering model for the BPP reads as follows:

$$\min_{\boldsymbol{y} \in \{0,1\}^{|\mathscr{S}|}} \left\{ \sum_{S \in \mathscr{S}} y_S : \sum_{S \in \mathscr{S}(j)} y_S \geq 1, \ \forall j \in N \right\}$$

A very strong ILP model with an **exponential** number of binary variables!

The fractional bin packing problem

The **lower bound** ζ provided by the optimal solution value of the LP relaxation is called the fractional bin packing number:

$$\zeta = \min_{y \geq 0} \left\{ \sum_{S \in \mathscr{S}} y_S : \quad \sum_{S \in \mathscr{S}(j)} y_S \geq 1, \ \ \forall j \in N \right\}$$

The difference between the optimal solution values of the set-covering model and those of the its LP relaxation is, typically, smaller or equal to 1:

modified integer round-up property (MIRUP) conjecture

The dual model reads as follows:

$$\zeta = \max_{\pi \geq 0} \left\{ \sum_{j \in N} \pi_j : \quad \sum_{i \in S} \pi_j \leq 1, \ \ \forall S \in \mathscr{S} \right\}$$

A LP model with an **exponential** number of constraints!

The fractional bin packing problem

The **lower bound** ζ provided by the optimal solution value of the LP relaxation is called the fractional bin packing number:

$$\zeta = \min_{y \geq 0} \left\{ \sum_{s \in \mathscr{S}} y_s : \quad \sum_{s \in \mathscr{S}(j)} y_s \geq 1, \ \ \forall j \in N \right\}$$

The difference between the optimal solution values of the set-covering model and those of the its LP relaxation is, typically, smaller or equal to 1:

modified integer round-up property (MIRUP) conjecture

The dual model reads as follows:

$$\zeta = \max_{\pi \geq 0} \left\{ \sum_{j \in N} \pi_j : \quad \sum_{i \in S} \pi_j \leq 1, \ \ \forall S \in \mathscr{S} \right\}$$

A LP model with an **exponential** number of constraints!

The fractional bin packing problem

The **lower bound** ζ provided by the optimal solution value of the LP relaxation is called the fractional bin packing number:

$$\zeta = \min_{\boldsymbol{y} \geq 0} \left\{ \sum_{S \in \mathscr{S}} y_S : \quad \sum_{S \in \mathscr{S}(j)} y_S \geq 1, \quad \forall j \in N \right\}$$

The difference between the optimal solution values of the set-covering model and those of the its LP relaxation is, typically, smaller or equal to 1:

modified integer round-up property (MIRUP) conjecture

The dual model reads as follows:

$$\zeta = \max_{\boldsymbol{\pi} \geq 0} \left\{ \sum_{j \in N} \pi_j : \quad \sum_{i \in S} \pi_j \leq 1, \quad \forall S \in \mathscr{S} \right\}$$

A LP model with an **exponential** number of constraints!

Primal-column generation – dual-constraint separation

The dual model containing a subset constraints is called **restricted master problem** (RMP):

$$\max_{\boldsymbol{\pi} \geq 0} \left\{ \sum_{j \in N} \pi_j : \quad \sum_{i \in S} \pi_j \leq 1, \ \ \forall S \in \tilde{\mathscr{S}} \right\}$$

Let π^* be an optimal dual solution, the **pricing problem** is:

$$\text{find} \ \ S^* \in \mathscr{S} \quad \text{such that} \quad \sum_{j \in S^*} \pi_j^* > 1$$

It is equivalent to the following **knapsack problem** (KP):

$$\max_{x \in \{0,1\}^n} \left\{ \sum_{j \in N} \pi_j^* \, x_j : \quad \sum_{j \in N} w_j \, x_j \leq C \right\}.$$

If its optimal solution value is > 1 then a violated dual constraint is found and added to the RMP.

Primal-column generation – dual-constraint separation

The dual model containing a subset constraints is called **restricted master problem** (RMP):

$$\max_{\boldsymbol{\pi} \geq 0} \left\{ \sum_{j \in N} \pi_j : \quad \sum_{i \in S} \pi_j \leq 1, \ \ \forall S \in \tilde{\mathscr{S}} \right\}$$

Let $\boldsymbol{\pi}^*$ be an optimal dual solution, the **pricing problem** is:

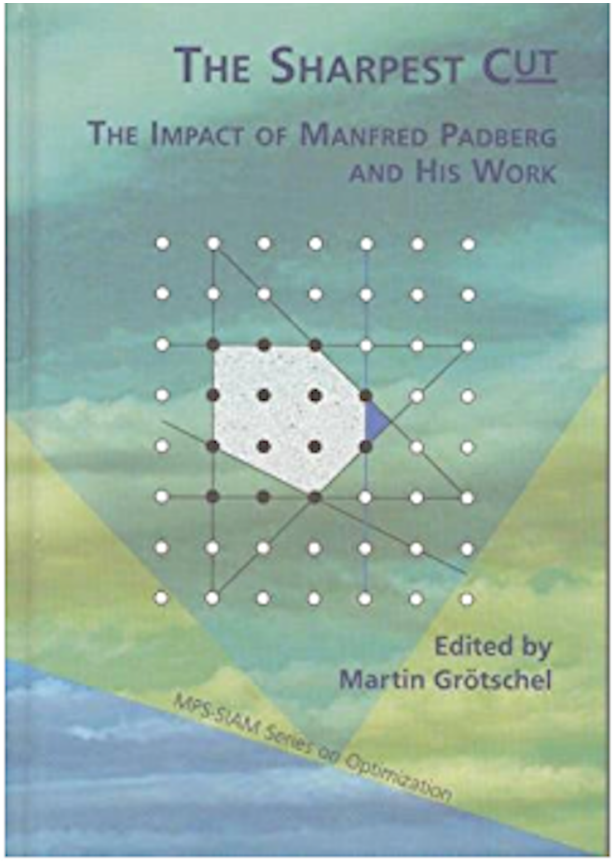
$$\text{find} \ \ S^* \in \mathscr{S} \quad \text{such that} \quad \sum_{j \in S^*} \pi_j^* > 1$$

It is equivalent to the following **knapsack problem** (KP):

$$\max_{x \in \{0,1\}^n} \left\{ \sum_{j \in N} \pi_j^* \, x_j : \quad \sum_{j \in N} w_j \, x_j \leq C \right\}.$$

If its optimal solution value is > 1 then a violated dual constraint is found and added to the RMP.

Separation of dual constraints



[1] Edited by Martin Grötschel.
The Sharpest Cut: The Impact of Manfred Padberg and His Work.
SIAM Series on Optimization, 2004.

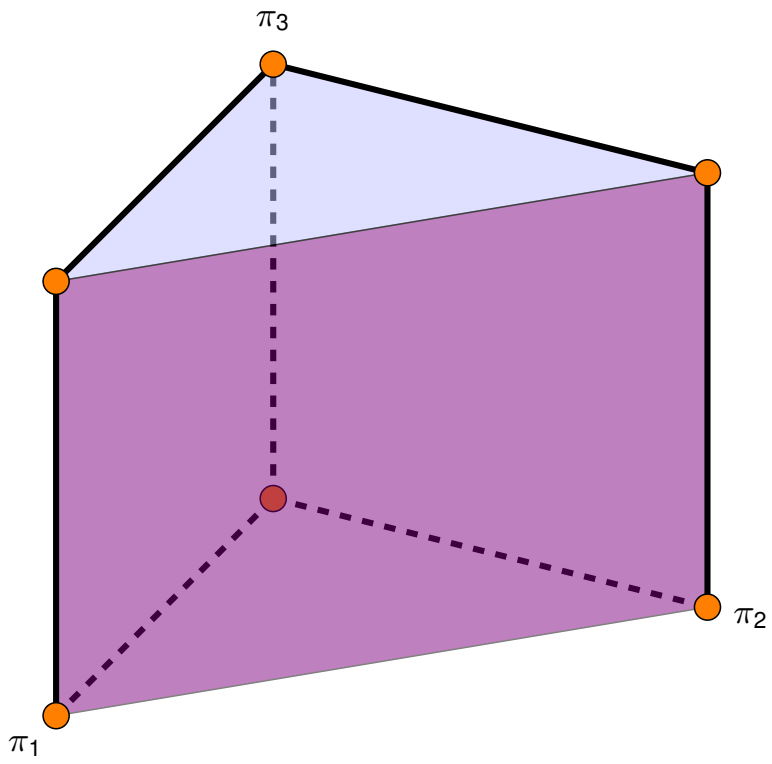
Example of dual constraints

$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_2 \leq 1 \right\}$$

$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_3 \leq 1 \right\}$$

Example of dual constraints

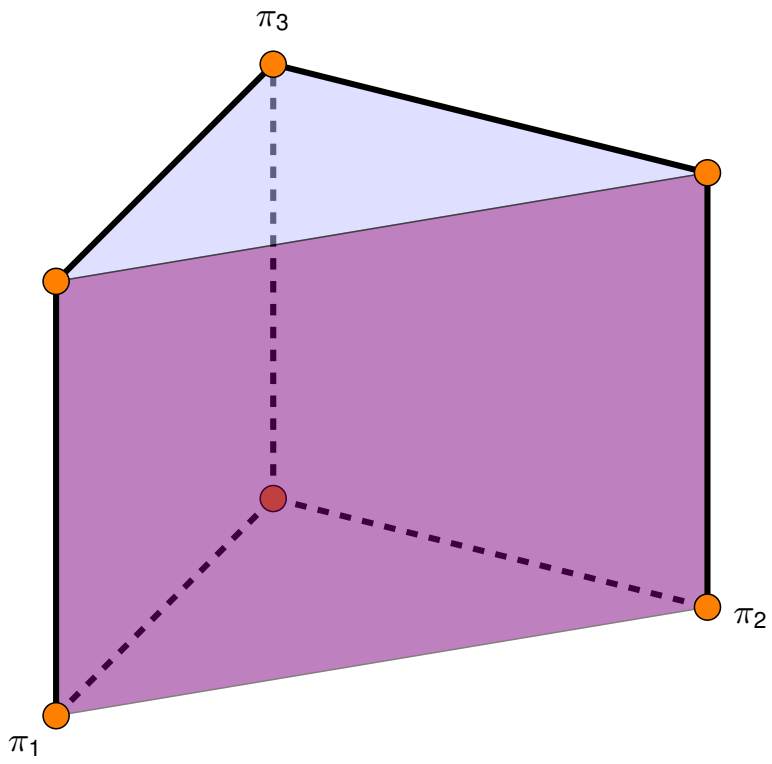
$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_2 \leq 1 \right\}$$



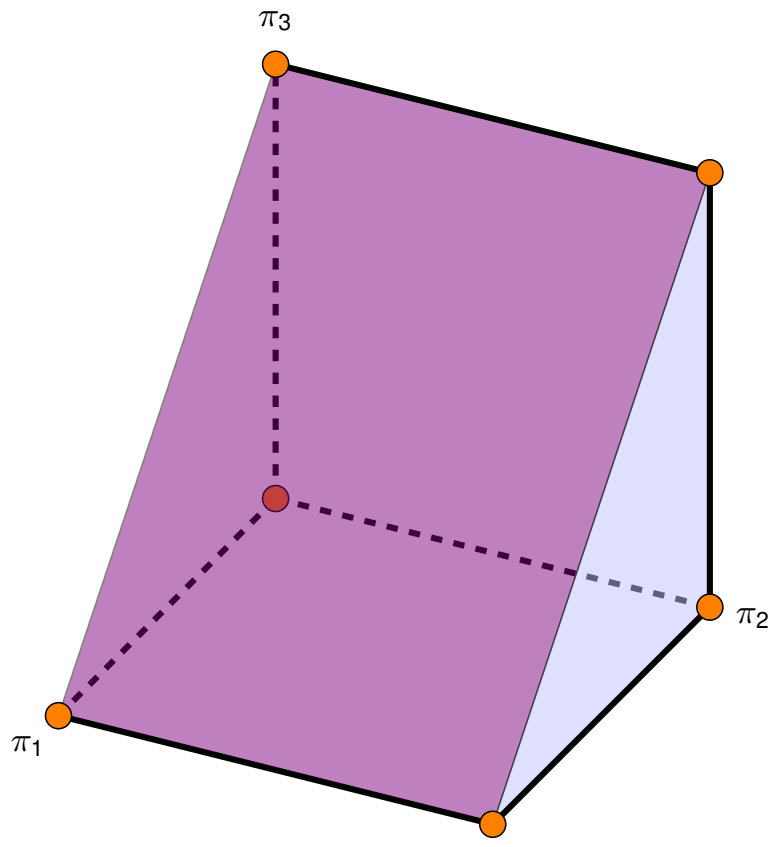
$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_3 \leq 1 \right\}$$

Example of dual constraints

$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_2 \leq 1 \right\}$$



$$\left\{ \boldsymbol{\pi} \in \mathbb{R}^3_+ : \quad \pi_1 + \pi_3 \leq 1 \right\}$$



Maximum dual-constraint violation

An optimal RMP solution:

$$\pi_1^* = \frac{7}{10},\ \pi_2^* = 1, \pi_3^* = 0$$

The pricing problem:

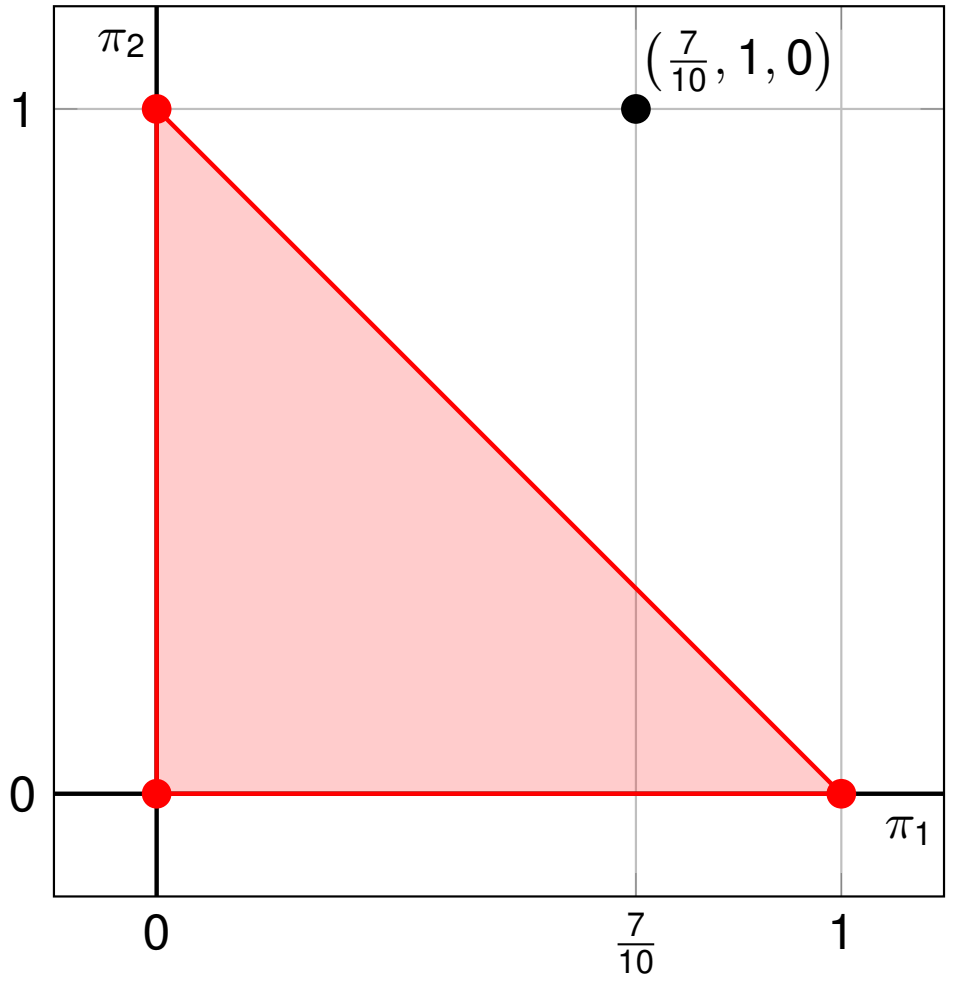
$$\max\ \frac{7}{10}\, x_1 + x_2$$

$$3\, x_1 + 2\, x_2 + x_3 \leq 5$$

$$x_1, x_2, x_3 \in \{0, 1\}$$

$S^* = \{1, 2\}$ and the dual constraint:

$$\pi_1 + \pi_2 \leq 1$$



Geometrical interpretation of the maximum violation

The **constraint violation** is the **minimum distance** of the point $\boldsymbol{\pi}^* \in \mathbb{R}^n$ from the supporting hyper-plane of the dual constraint **multiplied by the 2-norm of its coefficients**.

A dual constraint:

$$\sum_{j \in N} x_j \, \pi_j \leq 1 \text{ with } x_j \in \{0,1\}, \forall j \in N$$

its supporting hyperplane:

$$\sum_{j \in N} x_j \, \pi_j - 1 = 0$$

the euclidean distance:

$$d(\boldsymbol{\pi}^*, \boldsymbol{x}) = \frac{\left| \sum_{j \in N} x_j \, \pi_j^* - 1 \right|}{||\mathbf{x}||_2}$$

Geometrical interpretation of the maximum violation

The **constraint violation** is the **minimum distance** of the point $\boldsymbol{\pi}^* \in \mathbb{R}^n$ from the supporting hyper-plane of the dual constraint **multiplied by the 2-norm of its coefficients**.

A dual constraint:

$$\sum_{j \in N} x_j \, \pi_j \leq 1 \text{ with } x_j \in \{0, 1\}, \forall j \in N$$

its supporting hyperplane:

$$\sum_{j \in N} x_j \, \pi_j - 1 = 0$$

the euclidean distance:

$$d(\boldsymbol{\pi}^*, \boldsymbol{x}) = \frac{\left| \sum_{j \in N} x_j \, \pi_j^* - 1 \right|}{||\mathbf{x}||_2}$$

Geometrical interpretation of the maximum violation

The **constraint violation** is the **minimum distance** of the point $\pi^* \in \mathbb{R}^n$ from the supporting hyper-plane of the dual constraint **multiplied by the 2-norm of its coefficients**.

A dual constraint:

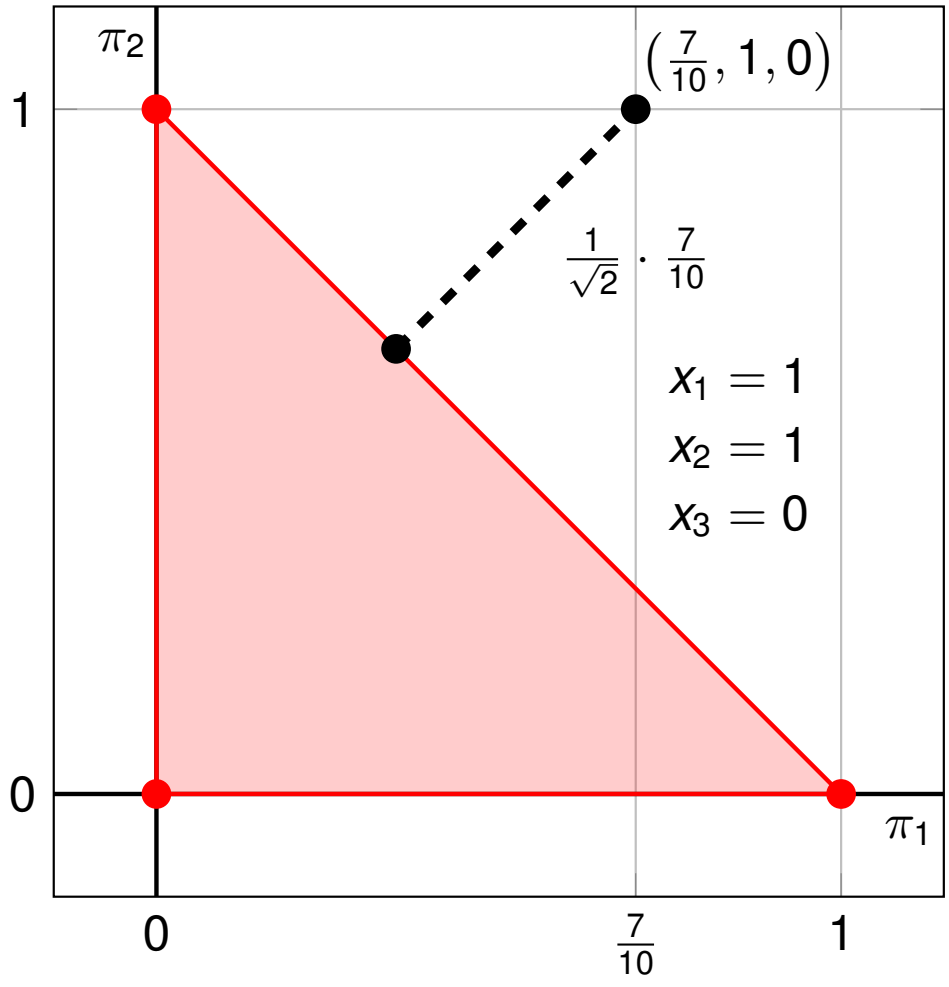
$$\sum_{j \in N} x_j \, \pi_j \leq 1 \text{ with } x_j \in \{0, 1\}, \forall j \in N$$

its supporting hyperplane:

$$\sum_{j \in N} x_j \, \pi_j - 1 = 0$$

the euclidean distance:

$$d(\pi^*, \boldsymbol{x}) = \frac{\left| \sum_{j \in N} x_j \, \pi_j^* - 1 \right|}{||\mathbf{x}||_2}$$



Sharp BPP columns

Sharp BPP columns – basic concepts

Rule of thumb

- The pricing problems are solved very efficiently by specialized dynamic programming algorithms
- Pattern of maximum violation correspond to columns with the largest reduced costs
- Maximal patterns lead to non-dominated dual constraints

... but we can do a bit more to speed up the convergence!

Sharp BPP columns – basic concepts

Rule of thumb

- The pricing problems are solved very efficiently by specialized dynamic programming algorithms
- Pattern of maximum violation correspond to columns with the largest reduced costs
- Maximal patterns lead to non-dominated dual constraints

... but we can do a bit more to speed up the convergence!

Sharp BPP columns – basic concepts

Rule of thumb

- The pricing problems are solved very efficiently by specialized dynamic programming algorithms
- Pattern of maximum violation correspond to columns with the largest reduced costs
- Maximal patterns lead to non-dominated dual constraints

... but we can do a bit more to speed up the convergence!

Sharp BPP columns – basic concepts

Rule of thumb

- The pricing problems are solved very efficiently by specialized dynamic programming algorithms
- Pattern of maximum violation correspond to columns with the largest reduced costs
- Maximal patterns lead to non-dominated dual constraints

... but we can do a bit more to speed up the convergence!

Maximal columns – non-dominated constraints

The pricing problem very often admit many different maximal columns even when restricting ourselves to columns of maximum reduced cost/violation.

- This is clear when π^* is sparse, which is often the case as, due to complementary slackness:

$$\pi_j^* = 0 \quad \text{whenever} \quad \sum_{S \in \mathscr{S}(j)} y_S > 1$$

The natural question is then:

should some of these maximal columns be preferred to the other ones?

Maximal columns – non-dominated constraints

The pricing problem very often admit many different maximal columns even when restricting ourselves to columns of maximum reduced cost/violation.

- This is clear when π^* is sparse, which is often the case as, due to complementary slackness:

$$\pi_j^* = 0 \quad \text{whenever} \quad \sum_{S \in \mathscr{S}(j)} y_S > 1$$

The natural question is then:

should some of these maximal columns be preferred to the other ones?

Three measures of maximality

1. Item-Weight

   The total weight of the items in the pattern $S \subseteq N$:

   $$g_w(x) = \sum_{j \in N} w_j \, x_j$$

   maximum weight $\sum_{j \in S} w_j$ implies minimum waste $C - \sum_{j \in S} w_j$

2. Item-Diversity

   The 1-norm distance $||x - \tilde{s}||_1$ between the column and the average $\tilde{s}$ of the previously generated columns plus a trade-off with the density:

   $$g_c(x) := ||x - \tilde{s}||_1 + \delta ||x||_1 = \sum_{j \in N} (3 - 2\tilde{s}_j) \, x_j + \sum_{j \in N} \tilde{s}_j$$

   Since the columns are binary and setting $\delta = 2$ (see Amaldi et al. 2014.)

3. Item-Density

   $$g_d(x) = ||x||_1 = \sum_{j \in N} x_j = |S|$$

Three measures of maximality

1. Item-Weight

   The total weight of the items in the pattern $S \subseteq N$:

   $$g_w(x) = \sum_{j \in N} w_j \, x_j$$

   maximum weight $\sum_{j \in S} w_j$ implies minimum waste $C - \sum_{j \in S} w_j$

2. Item-Diversity

   The 1-norm distance $||x - \tilde{s}||_1$ between the column and the average $\tilde{s}$ of the previously generated columns plus a trade-off with the density:

   $$g_c(x) := ||x - \tilde{s}||_1 + \delta ||x||_1 = \sum_{j \in N} (3 - 2\tilde{s}_j) \, x_j + \sum_{j \in N} \tilde{s}_j$$

   Since the columns are binary and setting $\delta = 2$ (see Amaldi et al. 2014.)

3. Item-Density

   $$g_d(x) = ||x||_1 = \sum_{j \in N} x_j = |S|$$

Three measures of maximality

1. Item-Weight

   The total weight of the items in the pattern $S \subseteq N$:

   $$g_w(x) = \sum_{j \in N} w_j \, x_j$$

   maximum weight $\sum_{j \in S} w_j$ implies minimum waste $C - \sum_{j \in S} w_j$

2. Item-Diversity

   The 1-norm distance $||x - \tilde{s}||_1$ between the column and the average $\tilde{s}$ of the previously generated columns plus a trade-off with the density:

   $$g_c(x) := ||x - \tilde{s}||_1 + \delta ||x||_1 = \sum_{j \in N} (3 - 2\tilde{s}_j) \, x_j + \sum_{j \in N} \tilde{s}_j$$

   Since the columns are binary and setting $\delta = 2$ (see Amaldi et al. 2014.)

3. Item-Density

   $$g_d(x) = ||x||_1 = \sum_{j \in N} x_j = |S|$$

Three measures of maximality

1. Item-Weight

   The total weight of the items in the pattern $S \subseteq N$:

   $$g_w(x) = \sum_{j \in N} w_j \, x_j$$

   maximum weight $\sum_{j \in S} w_j$ implies minimum waste $C - \sum_{j \in S} w_j$

2. Item-Diversity

   The 1-norm distance $||x - \tilde{s}||_1$ between the column and the average $\tilde{s}$ of the previously generated columns plus a trade-off with the density:

   $$g_c(x) := ||x - \tilde{s}||_1 + \delta ||x||_1 = \sum_{j \in N} (3 - 2\tilde{s}_j) \, x_j + \sum_{j \in N} \tilde{s}_j$$

   Since the columns are binary and setting $\delta = 2$ (see Amaldi et al. 2014.)

3. Item-Density

   $$g_d(x) = ||x||_1 = \sum_{j \in N} x_j = |S|$$

Example of different maximal columns

- Consider the following instance with $n = 6$ items and $C = 100$:

j	1	2	3	4	5	6
w_j	50	8	9	49	26	25

The RMP contains $S_j = \{j\}, \forall j \in N$ and $S_7 = \{1, 2, 3, 6\}$.

π_j^*	1	0	0	1	1	0
$\tilde{s}_j$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{2}{7}$
$3 - 2\,\tilde{s}_j$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{19}{7}$	$\frac{19}{7}$	$\frac{17}{7}$

- Maximal patterns of maximum reduced cost equal to 1:

pattern	density g_d	weight g_w	diversity g_c
$S_8 = \{1, 2, 3, 5\}$	4	93	10
$S_9 = \{2, 3, 4, 5\}$	4	92	$\frac{72}{7} \approx 10.28$
$S_{10} = \{4, 5, 6\}$	3	100	$\frac{55}{7} \approx 7.85$

Example of different maximal columns

- Consider the following instance with $n = 6$ items and $C = 100$:

j	1	2	3	4	5	6
w_j	50	8	9	49	26	25

The RMP contains $S_j = \{j\}, \forall j \in N$ and $S_7 = \{1, 2, 3, 6\}$.

π_j^*	1	0	0	1	1	0
$\tilde{s}_j$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{2}{7}$
$3 - 2\,\tilde{s}_j$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{19}{7}$	$\frac{19}{7}$	$\frac{17}{7}$

- Maximal patterns of maximum reduced cost equal to 1:

pattern	density g_d	weight g_w	diversity g_c
$S_8 = \{1, 2, 3, 5\}$	4	93	10
$S_9 = \{2, 3, 4, 5\}$	4	92	$\frac{72}{7} \approx 10.28$
$S_{10} = \{4, 5, 6\}$	3	100	$\frac{55}{7} \approx 7.85$

Example of different maximal columns

- Consider the following instance with $n = 6$ items and $C = 100$:

j	1	2	3	4	5	6
w_j	50	8	9	49	26	25

The RMP contains $S_j = \{j\}, \forall j \in N$ and $S_7 = \{1, 2, 3, 6\}$.

π_j^*	1	0	0	1	1	0
$\tilde{s}_j$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{2}{7}$
$3 - 2\,\tilde{s}_j$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{17}{7}$	$\frac{19}{7}$	$\frac{19}{7}$	$\frac{17}{7}$

- Maximal patterns of maximum reduced cost equal to 1:

pattern	density g_d	weight g_w	diversity g_c
$S_8 = \{1, 2, 3, 5\}$	4	93	10
$S_9 = \{2, 3, 4, 5\}$	4	92	$\frac{72}{7} \approx 10.28$
$S_{10} = \{4, 5, 6\}$	3	100	$\frac{55}{7} \approx 7.85$

Lexicographic dynamic programming pricing (LPP)

Multi-objective pricing problem with two objectives: the maximum reduced cost and one of the three maximality measures g_d, g_w and g_c.

$$\max_{x\in\{0,1\}^n} \operatorname{lex} \left\{ \big(f(x), g(x)\big) : \sum_{j\in N} w_j\, x_j \leq C \right\} \quad \text{where} \quad f(x) = \sum_{j\in N} \pi_j^*\, x_j$$

Lexicographic recursive formula (ϕ represents f and γ represents g):

$$\begin{pmatrix} \phi_j(s) \\ \gamma_j(s) \end{pmatrix} = \begin{cases} \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \succeq \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \\ \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \prec \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \end{cases}$$

$\phi_j(s)$ and $\gamma_j(s)$ are the values in terms of, respectively, f and g, of an optimal solution to the problem restricted to items in $\{1,\ldots,j\}$ and capacity $s \leq C$ ($c_j = 1$, $c_j = w_j$, and $c_j = 3 - 2\tilde{s}_j$ for g_d, g_w, and g_c).

Pseudopolynomial time complexity $O(n\,C)$ as the standard DP!

Lexicographic dynamic programming pricing (LPP)

Multi-objective pricing problem with two objectives: the maximum reduced cost and one of the three maximality measures g_d, g_w and g_c.

$$\max_{x\in\{0,1\}^n} \operatorname{lex} \left\{ \big(f(x), g(x)\big) : \sum_{j\in N} w_j\, x_j \le C \right\} \quad \text{where} \quad f(x) = \sum_{j\in N} \pi_j^*\, x_j$$

Lexicographic recursive formula (ϕ represents f and γ represents g):

$$\begin{pmatrix} \phi_j(s) \\ \gamma_j(s) \end{pmatrix} = \begin{cases} \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \succeq \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \\ \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \prec \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \end{cases}$$

$\phi_j(s)$ and $\gamma_j(s)$ are the values in terms of, respectively, f and g, of an optimal solution to the problem restricted to items in $\{1,\ldots,j\}$ and capacity $s \le C$ ($c_j = 1$, $c_j = w_j$, and $c_j = 3 - 2\tilde{s}_j$ for g_d, g_w, and g_c).

Pseudopolynomial time complexity $O(n\,C)$ as the standard DP!

Lexicographic dynamic programming pricing (LPP)

Multi-objective pricing problem with two objectives: the maximum reduced cost and one of the three maximality measures g_d, g_w and g_c.

$$\max_{x\in\{0,1\}^n} \operatorname{lex} \left\{ \left(f(x), g(x)\right) : \sum_{j\in N} w_j\, x_j \leq C \right\} \quad \text{where} \quad f(x) = \sum_{j\in N} \pi_j^*\, x_j$$

Lexicographic recursive formula (ϕ represents f and γ represents g):

$$\begin{pmatrix} \phi_j(s) \\ \gamma_j(s) \end{pmatrix} = \begin{cases} \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \succeq \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \\ \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} & \text{if } \begin{pmatrix} \phi_{j-1}(s) \\ \gamma_{j-1}(s) \end{pmatrix} \prec \begin{pmatrix} \phi_{j-1}(s-w_j)+\pi_j^* \\ \gamma_{j-1}(s-w_j)+c_j \end{pmatrix} \end{cases}$$

$\phi_j(s)$ and $\gamma_j(s)$ are the values in terms of, respectively, f and g, of an optimal solution to the problem restricted to items in $\{1,\ldots,j\}$ and capacity $s \leq C$ ($c_j = 1$, $c_j = w_j$, and $c_j = 3 - 2\tilde{s}_j$ for g_d, g_w, and g_c).

Pseudopolynomial time complexity $O(n\,C)$ as the standard DP!

Computational experience

Testbed BPP instances

class	# inst	n min	n max	C min	C max
Falkenauer T	80	60	501	1000	1000
Falkenauer U	80	120	1000	150	150
Hard 28	28	160	200	1000	1000
School 1	720	50	500	100	150
School 2	480	50	500	1000	1000
School 3	10	200	200	100000	100000
Schwerin 1	100	100	100	1000	1000
Schwerin 2	100	120	120	1000	1000
Wäscher	17	57	239	10000	10000

We discard 130 easy and small instances of class School 1 since they are all solved by generating less that 100 columns, thus obtaining a testbed of:

1475 instances

Testbed BPP instances

class	# inst	n min	n max	C min	C max
Falkenauer T	80	60	501	1000	1000
Falkenauer U	80	120	1000	150	150
Hard 28	28	160	200	1000	1000
School 1	720	50	500	100	150
School 2	480	50	500	1000	1000
School 3	10	200	200	100000	100000
Schwerin 1	100	100	100	1000	1000
Schwerin 2	100	120	120	1000	1000
Wäscher	17	57	239	10000	10000

We discard 130 easy and small instances of class School 1 since they are all solved by generating less that 100 columns, thus obtaining a testbed of:

1475 instances

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Configurations

The four CG algorithms are:

1. `DP-STD`: it employs the non-lexicographic DP algorithm by which the PP is solved so to guarantee the maximality of the resulting column.
2. `LEX-DENS`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_d(x)$ (i.e., using Density as second-level objective function).
3. `LEX-WEIGHT`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_w(x)$ (i.e., using Weight as second-level objective function).
4. `LEX-DIVER`: it employs the novel LEX-DP algorithm for solving the LPP with $g(x) = g_c(x)$ (i.e., using Diversity as second-level objective function).

They are all designed to guarantee the generation of a column of minimum reduced cost which is maximal.

Different optimal solutions?

class	different solutions [%]			last iter. diff. sol. [%]		
	Min	Avg	Max	Min	Avg	Max
Falkenauer T	39.17	57.63	71.34	40.83	69.94	99.16
Falkenauer U	9.08	21.46	52.03	58.22	83.85	99.52
Hard 28	16.48	25.42	41.85	45.16	86.14	99.85
School 1	0.00	46.71	100.00	0.00	80.25	99.96
School 2	1.19	22.43	93.42	14.00	42.80	99.87
School 3	28.00	30.54	32.98	38.25	87.02	98.82
Schwerin 1	14.00	18.30	22.22	22.11	32.92	99.48
Schwerin 2	11.57	18.29	25.69	21.63	36.24	98.62
Wäscher	3.01	10.81	25.48	12.74	37.62	99.45

Estimate of the number of times the pricing problem admits two different optimal solutions and of the last iteration in which two different optimal solutions are found.

Variation in the number of generated columns

class	# Cols	Percentage Column Variation		
	DP-STD	LEX-DIVER	LEX-WEIGHT	LEX-DENS
Falkenauer T	596.5	-4.0	-44.4	0.1
Falkenauer U	1117.0	0.6	-6.5	0.8
Hard 28	760.0	1.8	-5.7	3.8
School 1	611.1	-12.0	-4.2	4.3
School 2	485.1	-1.9	-12.7	-0.2
School 3	618.6	-4.9	-22.7	0.0
Schwerin 1	213.7	1.1	-9.2	-0.1
Schwerin 2	248.0	-5.0	-7.9	-0.1
Wäscher	540.2	0.0	-5.1	2.4

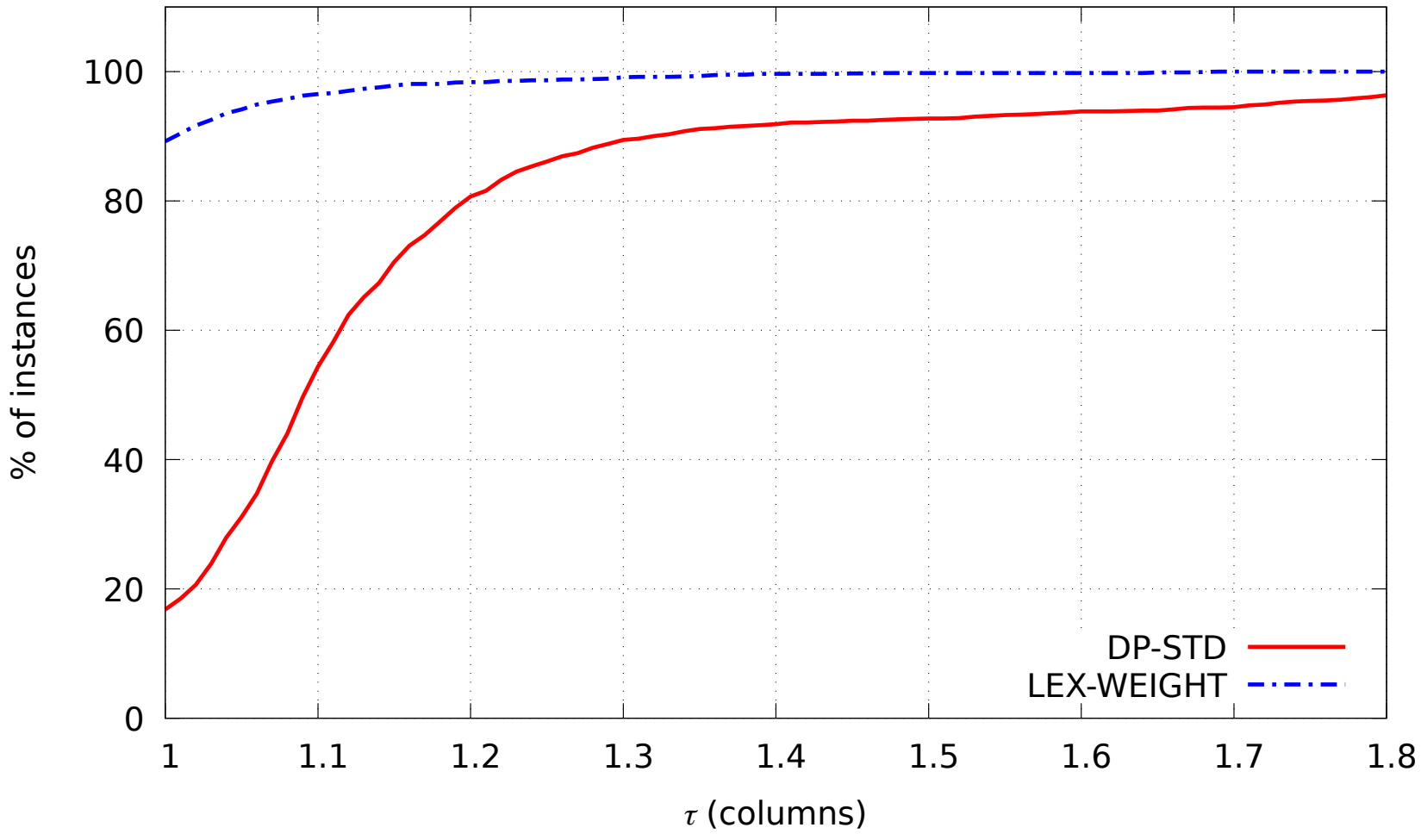
Average number of columns for DP-STD and percentage variation for LEX-DIVER, LEX-WEIGHT, and LEX-DENS per class of instances.

Variation in the computing time

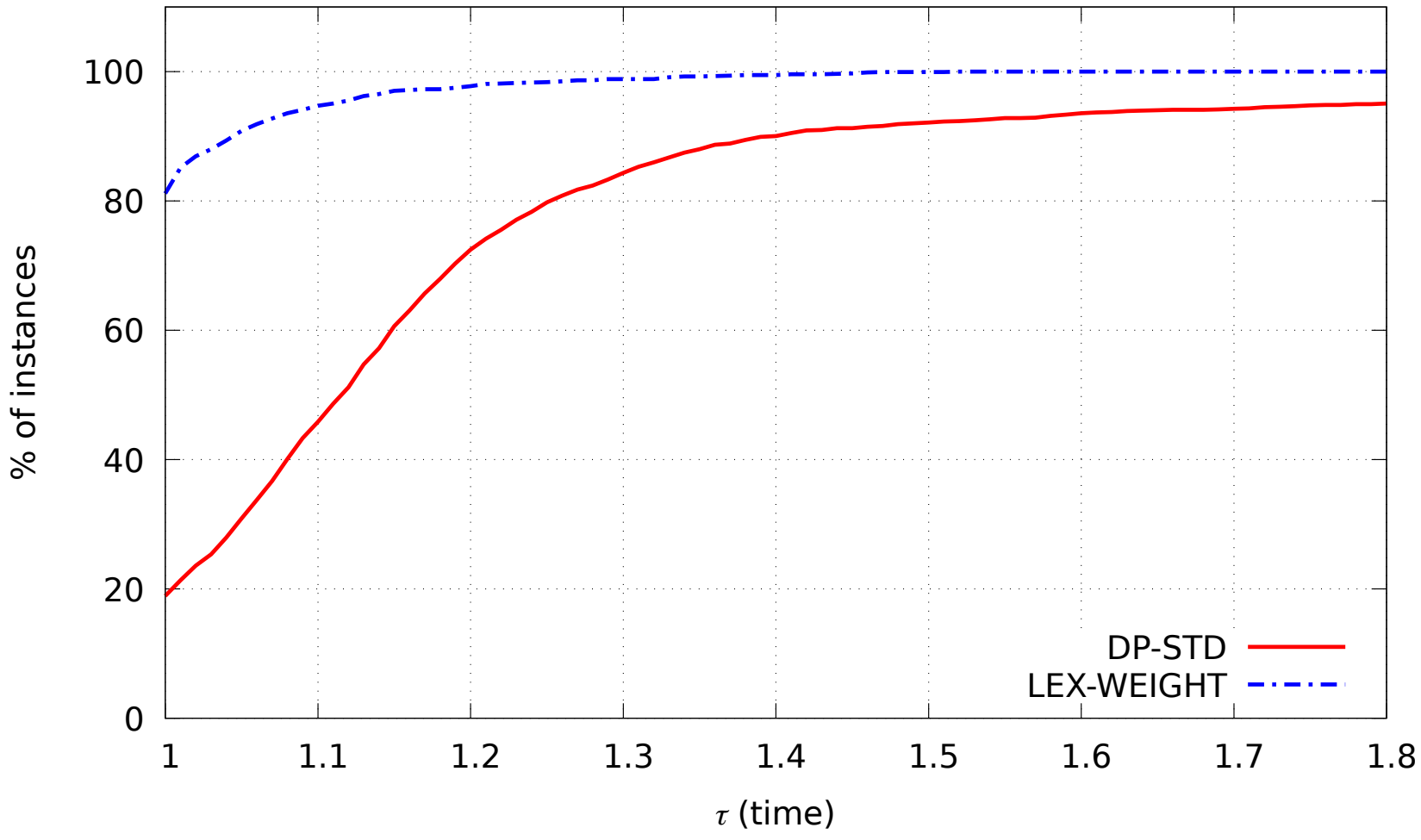
class	Time (s)	Percentage Time Variation		
	DP-STD	LEX-DIVER	LEX-WEIGHT	LEX-DENS
Falkenauer T	90.2	1.0	-58.2	2.5
Falkenauer U	766.7	-5.3	-4.2	-0.4
Hard 28	18.1	5.4	-4.6	7.0
School 1	508.0	0.6	-0.9	6.5
School 2	1672.2	-3.1	-14.8	0.9
School 3	405.9	-16.5	-35.3	0.4
Schwerin 1	12.7	7.7	-12.5	0.7
Schwerin 2	18.2	-6.1	-11.0	2.7
Wäscher	83.5	-1.6	-5.2	1.7

Total computing time (in seconds) for DP-STD and percentage variation for LEX-DIVER, LEX-WEIGHT, and LEX-DENS per class of instances.

Performance profile (columns)

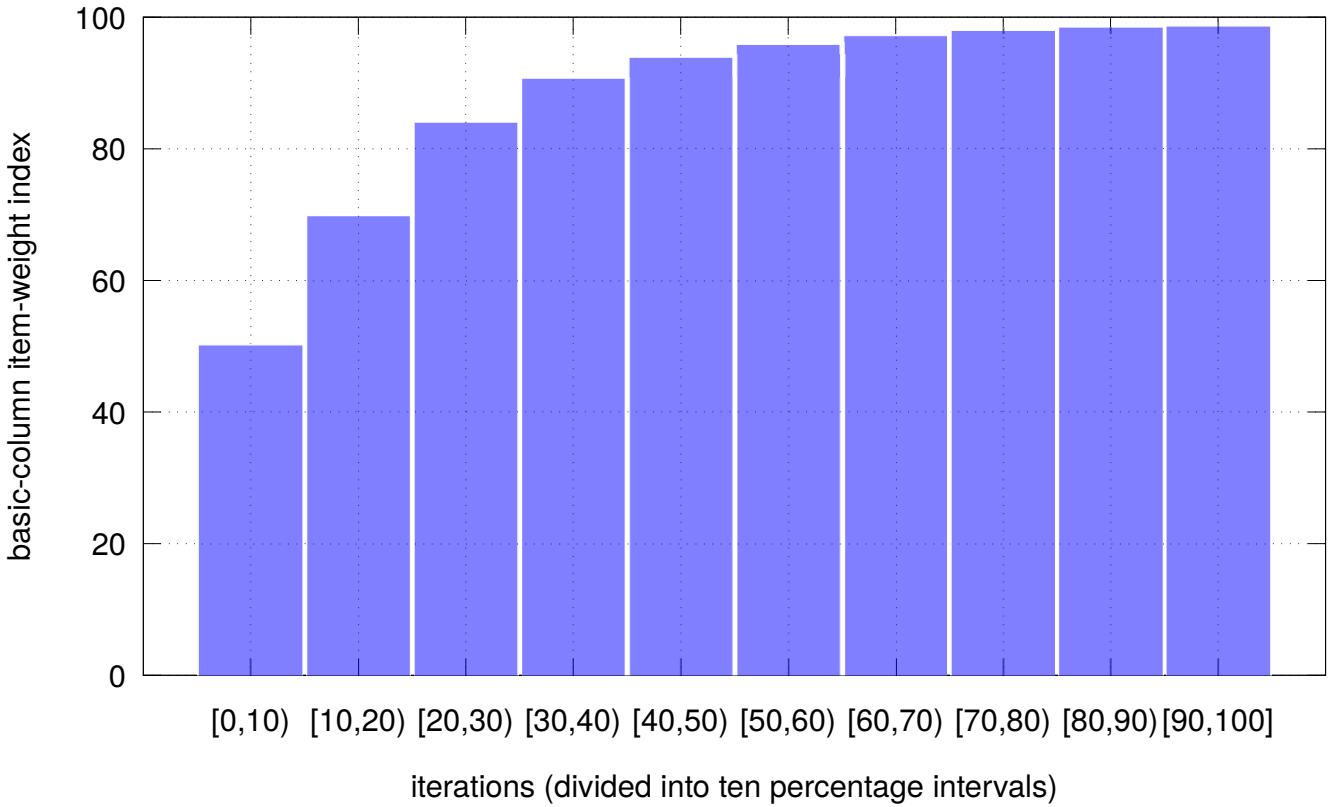


Performance profile (computing time)



Average filling of basic columns

For a given BPP instance and a given CG iteration, we compute the total item weight of the basic columns divided by the bin capacity, which we then average over the basic columns.



The basic-column item-weight index as a function of the number of iterations, subdivided into ten (percentage) intervals, obtained by running `DP-STD`.

Combination with a smoothing stabilization technique

	Perc. Column Variation		Perc. Time Variation	
class	DP−STD +SMOOTH	LEX−WEIGHT +SMOOTH	DP−STD +SMOOTH	LEX−WEIGHT +SMOOTH
Falkenauer T	-2.3	-44.7	-3.0	-51.8
Falkenauer U	-8.1	-14.6	-19.0	-25.7
Hard 28	-11.7	-16.4	-14.1	-19.0
School 1	-11.8	-15.5	-16.5	-20.2
School 2	-5.1	-16.4	-8.3	-21.1
School 3	-5.3	-26.9	-15.1	-38.3
Schwerin 1	-4.7	-11.9	-6.9	-15.3
Schwerin 2	-4.6	-10.7	-6.6	-14.7
Wäscher	-7.8	-11.2	-13.9	-16.1

Average number of columns and time percentage variation w.r.t. DP−STD obtained with DP−STD+SMOOTH and LEX−WEIGHT+SMOOTH per class of instances (see, e.g., Pessoa at al. 2018).

Conclusions

- We have proposed a lexicographic pricing problem for the FBPP which, among all the maximal columns of minimum reduced cost, generates one which maximizes one of three measures of maximality (density, weight, and diversity).
- Computational results on a large testbed of instances from the literature suggest that solving a lexicographic pricer is indeed advantageous, and that the adoption of the weight measure allows for a substantial reduction in the number of columns and computing time.

Open questions:

- Why the weight maximality measure works so well for the FBBP?
- Does this approach can be successfully used for other problems? Especially for problems in which the pricing problem is very time consuming.

Conclusions

- We have proposed a lexicographic pricing problem for the FBPP which, among all the maximal columns of minimum reduced cost, generates one which maximizes one of three measures of maximality (density, weight, and diversity).
- Computational results on a large testbed of instances from the literature suggest that solving a lexicographic pricer is indeed advantageous, and that the adoption of the weight measure allows for a substantial reduction in the number of columns and computing time.

Open questions:

- Why the weight maximality measure works so well for the FBBP?
- Does this approach can be successfully used for other problems? Especially for problems in which the pricing problem is very time consuming.

Conclusions

- We have proposed a lexicographic pricing problem for the FBPP which, among all the maximal columns of minimum reduced cost, generates one which maximizes one of three measures of maximality (density, weight, and diversity).
- Computational results on a large testbed of instances from the literature suggest that solving a lexicographic pricer is indeed advantageous, and that the adoption of the weight measure allows for a substantial reduction in the number of columns and computing time.

Open questions:

- Why the weight maximality measure works so well for the FBBP?
- Does this approach can be successfully used for other problems? Especially for problems in which the pricing problem is very time consuming.

Conclusions

- We have proposed a lexicographic pricing problem for the FBPP which, among all the maximal columns of minimum reduced cost, generates one which maximizes one of three measures of maximality (density, weight, and diversity).
- Computational results on a large testbed of instances from the literature suggest that solving a lexicographic pricer is indeed advantageous, and that the adoption of the weight measure allows for a substantial reduction in the number of columns and computing time.

Open questions:

- Why the weight maximality measure works so well for the FBBP?
- Does this approach can be successfully used for other problems? Especially for problems in which the pricing problem is very time consuming.

Conclusions

- We have proposed a lexicographic pricing problem for the FBPP which, among all the maximal columns of minimum reduced cost, generates one which maximizes one of three measures of maximality (density, weight, and diversity).
- Computational results on a large testbed of instances from the literature suggest that solving a lexicographic pricer is indeed advantageous, and that the adoption of the weight measure allows for a substantial reduction in the number of columns and computing time.

Open questions:

- Why the weight maximality measure works so well for the FBBP?
- Does this approach can be successfully used for other problems? Especially for problems in which the pricing problem is very time consuming.

Proposition

The optimal solutions of the FBPP and those of the following alternative problem coincide:

$$\arg\min_{\mathbf{y}\geq 0}\left\{\sum_{S\in\mathscr{S}}(C-\sum_{j\in S}w_j)\,y_S:\sum_{S\in\mathscr{S}(j)}y_S=1,\quad\forall j\in N\right\}.$$

Proof.

Due to the packing constraint, the objective function of the alternative problem is obtained by affine transformation of the objective function of the FBPP:

$$\begin{aligned}\sum_{S\in\mathscr{S}}(C-\sum_{j\in S}w_j)y_S &= C\sum_{S\in\mathscr{S}}y_S-\sum_{S\in\mathscr{S}}\sum_{j\in S}w_jy_S=\\ &= C\sum_{S\in\mathscr{S}}y_S-\sum_{j\in S}w_j\underbrace{\sum_{S\in\mathscr{S}:j\in S}y_S}_{=1}=\\ &= C\sum_{S\in\mathscr{S}}y_S-\sum_{j\in N}w_j.\end{aligned}$$

In an optimal FBBP solution the cover. constr. is satisfied as an equation. □

Proposition

Given any two patterns S_1, S_2 each of waste greater or equal than $\frac{C}{2}$, a solution to the FBPP with $y_{S_1} > 0$ and $y_{S_2} > 0$ cannot be optimal.

Proof.

The reason why such solution cannot be optimal is that, having waste greater or equal than $\frac{C}{2}$, the two patterns can be merged into a new pattern $S' := S_1 \cup S_2$.

Indeed, letting

$$y_{S'} = \max\{y_{S_1}, y_{S_2}\}, \quad y_{S_1} = 0 \ \text{ and } \ y_{S_2} = 0$$

we obtain another feasible solution with an objective function value smaller than that of the original one by

$$y_{S_1} + y_{S_2} - \max\{y_{S_1}, y_{S_2}\} > 0$$

□

References

[1] E. G. Coffman Jr, J. Csirik, G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: survey and classification, in: Handbook of combinatorial optimization, Springer, 2013, pp. 455–531.

[2] J. Valério de Carvalho, LP models for bin packing and cutting stock problems, European Journal of Operational Research 141 (2) (2002) 253–273.

[3] F. Clautiaux, C. Alves, J. Valério de Carvalho, A survey of dual-feasible and superadditive functions, Annals of Operations Research 179 (1) (2010) 317–342.

[4] M. Delorme, M. Iori, S. Martello, Bin packing and cutting stock problems: Mathematical models and exact algorithms, European Journal of Operational Research 255 (1) (2016) 1 – 20.

[5] G. Belov, G. Scheithauer, A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting, European Journal of Operational Research 171 (1) (2006) 85 – 106.

References

[1] A. Caprara, M. Dell'Amico, J. C. Díaz-Díaz, M. Iori, R. Rizzi, Friendly bin packing instances without integer round-up property, Mathematical Programming 150 (1) (2015) 5–17.

[2] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, New York, 1990.

[3] J. Desrosiers, M. E. Lübbecke, A Primer in Column Generation, Springer US, Boston, MA, 2005, pp. 1–32.

[4] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Stabilized column generation, Discrete Mathematics 194 (1) (1999) 229 – 237.

[5] P. Wentges, Weighted dantzig-wolfe decomposition for linear mixed-integer programming, International Transactions in Operational Research 4 (2) (1997) 151–162.

References

[1] J. Valério de Carvalho, Using extra dual cuts to accelerate column generation, INFORMS Journal on Computing 17 (2) (2005) 175–182.

[2] A. Frangioni, B. Gendron, A stabilized structured dantzig–wolfe decomposition method, Mathematical Programming 140 (1) (2013) 45–76.

[3] H. M. B. Amor, J. Desrosiers, A. Frangioni, On the choice of explicit stabilizing terms in column generation, Discrete Applied Mathematics 157 (6) (2009) 1167 – 1184.

[4] A. Pessoa, R. Sadykov, E. Uchoa, F. Vanderbeck, Automation and combination of linear-programming based stabilization techniques in column generation, INFORMS Journal on Computing 30 (2) (2018) 339–360.

[5] L. Wei, Z. Luo, R. Baldacci, A. Lim, A new branch-and-price-and-cut algorithm for one-dimensional bin packing problems, Accepted for publication on INFORMS Journal on Computing (2018) 1–32.

[6] F. Vanderbeck, Implementing mixed integer column generation, in: Column generation, Springer, 2005, pp. 331–358.

References

[1] M. Fischetti, A. Lodi, Optimizing over the first chvátal closure, Mathematical Programming 110 (1) (2007) 3–20.

[2] E. Balas, A. Saxena, Optimizing over the split closure, Mathematical Programming 113 (2) (2008) 219–240.

[3] A. Zanette, M. Fischetti, E. Balas, Lexicography and degeneracy: can a pure cutting plane algorithm work?, Mathematical programming 130 (1) (2011) 153–176.

[4] E. Amaldi, S. Coniglio, S. Gualandi, Coordinated cutting plane generation via multi-objective separation, Mathematical Programming 143 (1-2) (2014) 87–110.

[5] L.-M. Rousseau, Stabilization issues for constraint programming based column generation, in: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, Springer, 2004, pp. 402–408.

[6] M. E. Lübbecke, J. Desrosiers, Selected topics in column generation, Operations research 53 (6) (2005) 1007–1023.

References

[1] S. Gualandi, F. Malucelli, Exact solution of graph coloring problems via constraint programming and column generation, INFORMS Journal on Computing 24 (1) (2012) 81–100.

[2] E. Amaldi, S. Coniglio, S. Gualandi, Improving cutting plane generation with 0-1 inequalities by bi-criteria separation, in: International Symposium on Experimental Algorithms, Springer, 2010, pp. 266–275.

[3] T. Achterberg, SCIP: solving constraint integer programs, Math. Program. Comput. 1 (1) (2009) 1–41.

[4] E. Falkenauer, A hybrid grouping genetic algorithm for bin packing, J. Heuristics 2 (1) (1996) 5–30.

[5] J. Shoenfiled, Fast, exact solution of open bin packing problems without linear programming, Technical report, US Army Space and Missile Defence Command, Huntsville, Alabama, USA.

References

[1] A. Scholl, R. Klein, C. Jürgens, Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem, Computers & Operations Research 24 (7) (1997) 627 – 645.

[2] P. Schwerin, G. Wäscher, A new lower bound for the bin-packing problem and its integration into MTP, Martin-Luther-Univ. Alle-Wittenberg. Wirtschaftswiss. Fak.

[3] G. Wäscher, T. Gau, Heuristics for the integer one-dimensional cutting stock problem: A computational study, Operations-Research-Spektrum 18 (3) (1996) 131–144.

[4] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical programming 91 (2) (2002) 201–213.

[5] A. A. Farley, A note on bounding a class of linear programming problems, including cutting stock problems, Operations Research 38 (5) (1990) 922–923.

[6] H. B. Amor, J. V. de Carvalho, Cutting stock problems, in: Column generation, Springer, 2005, pp. 131–161.

References

[1] S. Coniglio, M. Tieves, On the generation of cutting planes which maximize the bound improvement, in: International Symposium on Experimental Algorithms, Springer, 2015, pp. 97–109.

[2] S. S. Dey, M. Molinaro, Q. Wang, Approximating polyhedra with sparse inequalities, Mathematical Programming 154 (1-2) (2015) 329–352.

[3] S. S. Dey, M. Molinaro, Theoretical challenges towards cutting-plane selection, Mathematical Programming 170 (1) (2018) 237–266.

[4] F. Furini, I. Ljubić, M. Sinnl, An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem, European Journal of Operational Research 262 (2) (2017) 438–448.