Pseudo-polynomial formulations for bin packing and cutting stock problems

Maxence Delorme[(1)] Manuel Iori[(2)]

(1) DEI, University of Bologna, Italy,
(2) DISMI, University of Modena and Reggio Emilia, Italy

COLGEN 2016 – Búzios

Outline

Bin Packing and Cutting Stock

In this talk we study Cutting & Packing (C&P) problems from a computational point of view

We focus on two (related) fundamental one-dimensional C&P problems:

- *Bin Packing Problem* (BPP): given n items having width w_j for $j = 1, \ldots, n$, and bins of capacity c, pack the items into the minimum number of bins
- *Cutting Stock Problem* (CSP): given m item types, each consisting of a demand d_j of items of width w_j, and stocks of length c, cut the items from the minimum number of stocks

Pseudo-polynomial formulations

Pseudo-polynomial formulations (*position-indexed* formulations, *capacity-indexed* formulations, *time-indexed* formulations, . . .) model CSP and BPP instances by taking care of the *position* in which the item is cut from/packed into the bin

- good lower bounds
- number of variables and constraints grows not only with n but also with c

Intensively studied for C&P problems from the 1960s:

- as a basis for branch-and-price algorithms
- as stand-alone models for optimal problem solution

Pseudo-polynomial formulations

Pseudo-polynomial formulations (*position-indexed* formulations, *capacity-indexed* formulations, *time-indexed* formulations, ...) model CSP and BPP instances by taking care of the *position* in which the item is cut from/packed into the bin

- good lower bounds
- number of variables and constraints grows not only with n but also with c

Intensively studied for C&P problems from the 1960s:

- as a basis for branch-and-price algorithms
- as stand-alone models for optimal problem solution

Evolution of software for MILP

Pseudo-polynomial formulations are by far the main mathematical model in multi-dimensional C&P since *Beasley (Operations Research, 1985)*

For the 1-dim, case, from *Delorme, Iori and Martello (EJOR, forthcoming)*:

- n. of selected CSP instances solved to optimality (average seconds, 2.66 GHz)

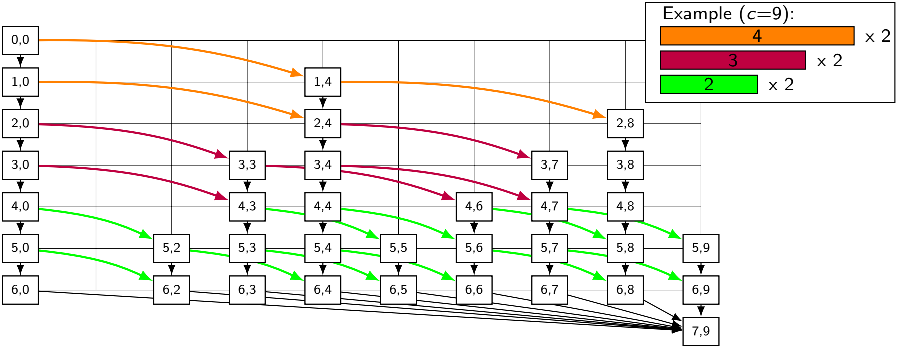
using the ARC-FLOW formulation by *Valério de Carvalho (Annals of OR, 1999)* and different CPLEX versions

	CPLEX	6.0	7.0	8.0	9.0	10.0	11.0	12.1	12.6
sec	inst	(1998)	(1999)	(2002)	(2003)	(2006)	(2007)	(2009)	(2013)
600	20	13 (366)	10 (420)	5 (570)	17 (268)	19 (162)	**20 (65)**	19 (117)	**20 (114)**
3600	20	16 (897)	15 (1210)	15 (2009)	**20 (343)**	**20 (186)**	**20 (65)**	19 (267)	**20 (114)**

Dynamic-Programming Flow Formulation (DP-FLOW)

Formally introduced for BPP only by *Cambazard and O'Sullivan (LNCS, 2010)*
Idea dates back to (at least) *Wolsey (Annals of Discrete Mathematics, 1977)*

- DP table: a BPP item j is an horizontal layer, a weight d is a vertical layer
- $x_{j,d,j+1,e}$ = number of times arc $((j,d),(j+1,e))$ is selected



Dynamic-Programming Flow Formulation (DP-FLOW)

Formally introduced for BPP only by *Cambazard and O'Sullivan (LNCS, 2010)*
Idea dates back to (at least) *Wolsey (Annals of Discrete Mathematics, 1977)*

- DP table: a BPP item j is an horizontal layer, a weight d is a vertical layer
- $x_{j,d,j+1,e}$ = number of times arc $((j,d),(j+1,e))$ is selected

$$\min \quad z$$

$$\sum_{((j,d),(j+1,e))\in\delta^+((j,d))} x_{j,d,j+1,e} - \sum_{((j-1,e),(j,d))\in\delta^-((j,d))} x_{j-1,e,j,d} = \begin{cases} z & \text{if } (j,d)=(0,0) \\ -z & \text{if } (j,d)=(n+1,c) \\ 0 & \text{otherwise} \end{cases}$$

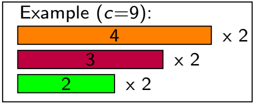
$$\sum_{((j-1,d),(j,d+w_j))\in A} x_{j-1,d,j,d+w_j} = 1 \qquad j=1,\dots,n$$

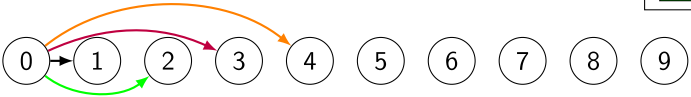
$$x_{j,d,j+1,e} \geq 0 \text{ and integer} \qquad ((j,d),(j+1,e)) \in A$$

Aggregate Flow Formulation (ARC-FLOW)

Formally introduced for CSP by *Valério de Carvalho (Annals of OR, 1999)* as a basis for a B&P

- all DP horizontal layers are "aggregated" into a unique layer
- x_{de} = n° of times arc (d, e) is chosen (item of weight $e-d$ or empty space)

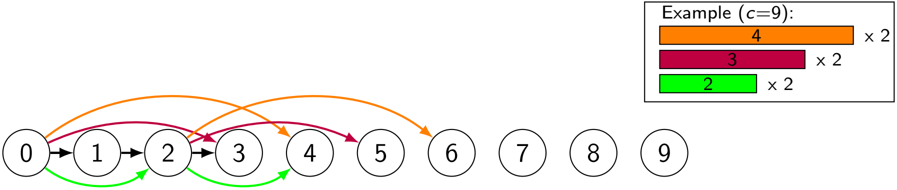




Aggregate Flow Formulation (ARC-FLOW)

Formally introduced for CSP by *Valério de Carvalho (Annals of OR, 1999)* as a basis for a B&P

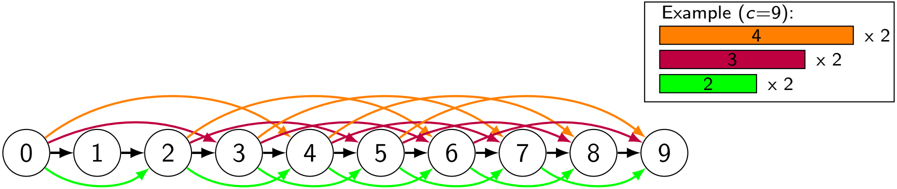
- all DP horizontal layers are "aggregated" into a unique layer
- x_{de} = n° of times arc (d, e) is chosen (item of weight $e - d$ or empty space)



Aggregate Flow Formulation (ARC-FLOW)

Formally introduced for CSP by *Valério de Carvalho (Annals of OR, 1999)* as a basis for a B&P

- all DP horizontal layers are “aggregated” into a unique layer
- x_{de} = n° of times arc (d, e) is chosen (item of weight $e-d$ or empty space)

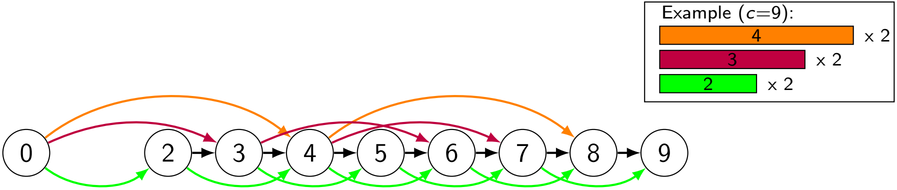


Full graph (27 arcs)

Aggregate Flow Formulation (ARC-FLOW)

Formally introduced for CSP by *Valério de Carvalho (Annals of OR, 1999)* as a basis for a B&P

- all DP horizontal layers are "aggregated" into a unique layer
- x_{de} = n$^\circ$ of times arc (d, e) is chosen (item of weight $e-d$ or empty space)



Reduced graph (19 arcs)

(remove redundant arcs, sort items by non-increasing width)

Aggregate Flow Formulation (ARC-FLOW)

Formally introduced for CSP by *Valério de Carvalho (Annals of OR, 1999)* as a basis for a B&P

- all DP horizontal layers are "aggregated" into a unique layer
- x_{de} = n° of times arc (d, e) is chosen (item of weight $e-d$ or empty space)

$$\min \quad z$$

$$-\sum_{(d,e)\in\delta^-(e)} x_{de} + \sum_{(e,f)\in\delta^+(e)} x_{ef} = \begin{cases} z & \text{if } e = 0 \\ -z & \text{if } e = c \\ 0 & \text{if } e = 1,\dots,c-1 \end{cases}$$

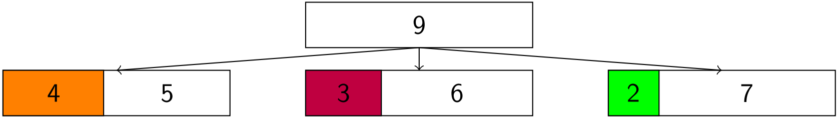
$$\sum_{(d,d+w_i)\in A} x_{d,d+w_i} \geq d_i \qquad i = 1,\dots,m$$

$$x_{de} \geq 0 \text{ and integer} \qquad (d,e) \in A$$

ONE-CUT

Proposed by *Rao (Journal of the Computer Society of India, 1976)* and *Dyckhoff (Operations Research, 1981)*

- y_{pq}= number of times a piece of width p is cut into a left piece (item) of width q and a right piece (residual) of width $p - q$

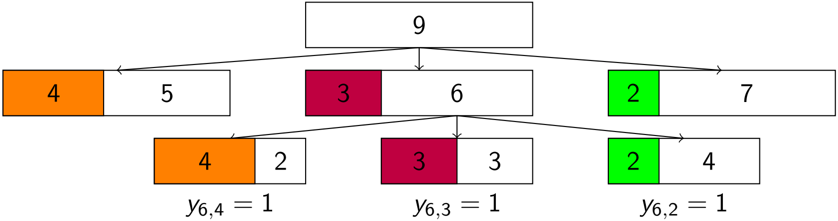


$y_{9,4} = 1$ $y_{9,3} = 1$ $y_{9,2} = 1$

ONE-CUT

Proposed by *Rao (Journal of the Computer Society of India, 1976)* and *Dyckhoff (Operations Research, 1981)*

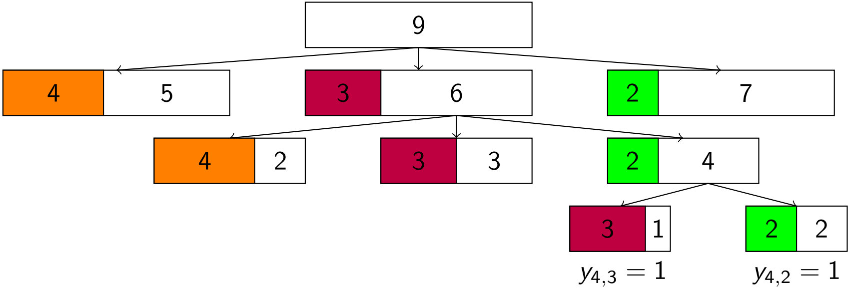
- y_{pq}= number of times a piece of width p is cut into a left piece (item) of width q and a right piece (residual) of width $p-q$



ONE-CUT

Proposed by *Rao (Journal of the Computer Society of India, 1976)* and *Dyckhoff (Operations Research, 1981)*

- y_{pq}= number of times a piece of width p is cut into a left piece (item) of width q and a right piece (residual) of width $p-q$



ONE-CUT

Proposed by *Rao (Journal of the Computer Society of India, 1976)* and *Dyckhoff (Operations Research, 1981)*

- y_{pq}= number of times a piece of width p is cut into a left piece (item) of width q and a right piece (residual) of width $p - q$

$$
\begin{aligned}
\text{min} \quad & z = \sum_{q \in W} y_{cq} & \\
\text{s.t.} \quad & \sum_{p \in A(q)} y_{pq} + \sum_{p \in B(q)} y_{p+q,p} \geq d_q + \sum_{r \in C(q)} y_{qr} & q \in (W \cup R) \backslash \{c\} \\
& y_{pq} \geq 0 \text{ and integer} & p \in R, q \in W, p > q
\end{aligned}
$$

Computational results for BPP/CSP

From *Delorme, Iori and Martello (EJOR, forthcoming)*:

- n. of instances solved to optimality in one minute (average CPU seconds) for number of items n

n	inst.	Pseudo-polynomial formulations DP-FLOW	ONE-CUT	ARCFLOW	Branch(-and-cut)-and-price SCIP-BPP	VANCE	BELOV
50	165	**165 (0.5)**	**165 (0.1)**	**165 (0.1)**	**165 (0.9)**	**165 (0.1)**	**165 (0.1)**
100	271	**271 (5.0)**	**271 (0.8)**	**271 (0.3)**	**271 (4.6)**	**271 (0.1)**	**271 (0.1)**
200	359	292 (21.0)	358 (2.4)	**359 (0.8)**	293 (22.6)	358 (1.1)	**359 (0.1)**
300	393	243 (33.9)	385 (4.5)	391 (2.0)	155 (44.1)	387 (4.3)	**393 (0.1)**
400	425	193 (42.4)	408 (5.1)	421 (3.0)	114 (49.8)	416 (9.3)	**425 (0.2)**
500	414	169 (44.8)	394 (6.3)	402 (4.0)	69 (55.1)	394 (19.2)	**414 (0.2)**
750	433	120 (52.6)	401 (7.8)	415 (6.0)	22 (59.5)	99 (50.4)	**433 (0.4)**
1000	441	67 (56.4)	407 (8.1)	416 (6.8)	0 (60.0)	62 (52.4)	**441 (0.7)**
Total	2901	1520 (36.7)	2789 (5.0)	2840 (3.3)	1089 (42.4)	2152 (20.3)	**2901 (0.2)**

ONE-CUT and ARC-FLOW somehow competitive up to n=200

Computational results for BPP/CSP

From *Delorme, Iori and Martello (EJOR, forthcoming)*:

- n. of instances solved to optimality in one minute (average CPU seconds) for bin capacity c

		Pseudo-polynomial formulations			Branch(-and-cut)-and-price		
c	inst.	DPFLOW	ONECUT	ARCFLOW	SCIP-BPP	VANCE	BELOV
50	223	205 (13.6)	**223 (0.1)**	**223 (0.1)**	86 (43.0)	162 (20.2)	**223 (0.1)**
75	240	208 (19.5)	**240 (0.1)**	**240 (0.1)**	96 (42.0)	176 (20.2)	**240 (0.1)**
100	234	185 (26.4)	**234 (0.1)**	**234 (0.1)**	89 (42.8)	172 (20.3)	**234 (0.1)**
120	241	168 (29.6)	**241 (0.1)**	**241 (0.1)**	91 (42.2)	181 (19.7)	**241 (0.1)**
125	251	174 (30.9)	**251 (0.1)**	**251 (0.1)**	101 (41.0)	192 (19.2)	**251 (0.1)**
150	240	143 (34.9)	**240 (0.1)**	**240 (0.2)**	95 (41.8)	181 (19.5)	**240 (0.1)**
200	246	127 (39.3)	**246 (0.3)**	**246 (0.3)**	99 (40.8)	184 (20.5)	**246 (0.1)**
300	237	79 (45.6)	**237 (1.5)**	**237 (1.3)**	80 (44.4)	172 (21.9)	**237 (0.1)**
400	245	71 (47.4)	**245 (3.9)**	**245 (2.8)**	95 (41.6)	184 (20.3)	**245 (0.2)**
500	243	56 (49.2)	241 (8.6)	242 (5.1)	77 (44.8)	179 (20.7)	**243 (0.3)**
750	249	55 (49.9)	211 (18.6)	229 (11.2)	91 (42.4)	183 (21.0)	**249 (0.8)**
1000	252	49 (51.5)	180 (25.3)	212 (17.6)	89 (42.5)	186 (20.7)	**252 (1.3)**
Total	2901	1520 (36.7)	2789 (5)	2840 (3.3)	1089 (42.4)	2152 (20.3)	**2901 (0.2)**

ONE-CUT and ARC-FLOW somehow competitive up to c=400

Equivalence of the formulations

All proposed formulation shave the same continuous lower bound value as that provided by the *Gilmore and Gomory (Operations Research, 1963)* model (GG)

Proposition (*Valério de Carvalho (Annals of OR, 1999)*)

ARC-FLOW is equivalent to GG

Proposition

DP-FLOW is equivalent to ARC-FLOW

Proof Skipped

Proposition

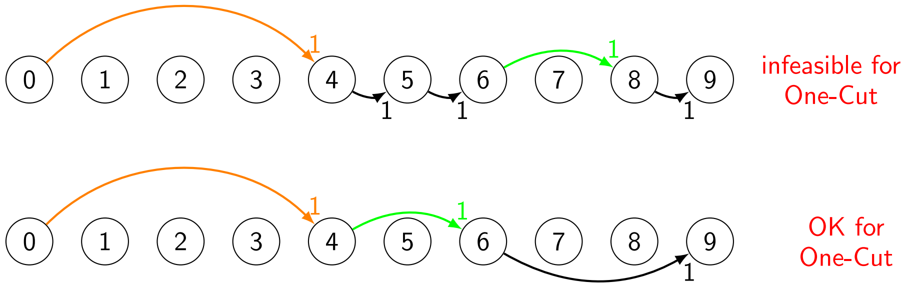
ONE-CUT is equivalent to ARC-FLOW

Proof based on decomposition of non-negative flows into paths and cycles

ARC-FLOW is included in ONE-CUT

We use an LP ARC-FLOW solution $[\bar{x}_{de}]$ to build an LP ONE-CUT solution $[\bar{y}_{pq}]$

First step, we transform ARC-FLOW into an equivalent Normal ARC-FLOW by replacing, for any d, original loss arcs $(d, d+1)$ with (d, c)



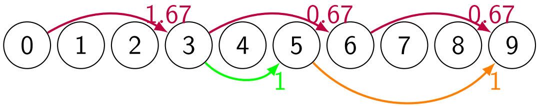
ARC-FLOW is included in ONE-CUT

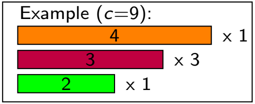
We use an LP ARC-FLOW solution $[\bar{x}_{de}]$ to build an LP ONE-CUT solution $[\bar{y}_{pq}]$

Second step, we build $[\bar{y}_{pq}]$ using the normal Arc-Flow solution as:

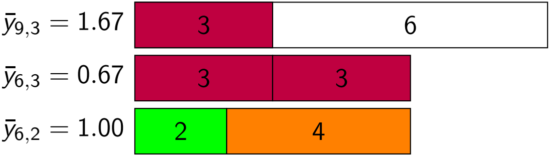
$$\bar{y}_{c-d,e-d} = \bar{x}_{de} \quad \forall (d,e) \in A, e \neq c$$

ARC-FLOW LP solution ($z = 1.67$):





Resulting ONE-CUT LP solution:



ONE-CUT is included in ARC-FLOW

We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

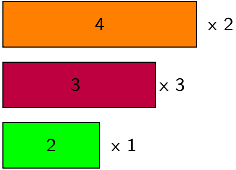
Set all $\bar{x}_{d,e} = 0$;
For all $\bar{y}_{p,q} > 0$ do
  $\bar{x}_{c-p,c-p+q} += \bar{y}_{p,q}$;
  $\bar{x}_{c-p+q,c} += \bar{y}_{p,q}$;
  if $p \neq c$
    $\bar{x}_{c-p,c} -= \bar{y}_{p,q}$;
  endif
end do;

ONE-CUT is included in ARC-FLOW

We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

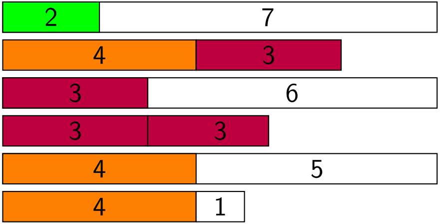
Set all $\bar{x}_{d,e} = 0$;
For all $\bar{y}_{p,q} > 0$ do
  $\bar{x}_{c-p,c-p+q} \mathrel{+}= \bar{y}_{p,q}$;
  $\bar{x}_{c-p+q,c} \mathrel{+}= \bar{y}_{p,q}$;
  if $p \neq c$
    $\bar{x}_{c-p,c} \mathrel{-}= \bar{y}_{p,q}$;
  endif
end do;

Example (c=9):



LP solution for ONE-CUT ($z = 2.17$):

$\bar{y}_{9,2} = 1.00$
$\bar{y}_{7,4} = 1.00$
$\bar{y}_{9,3} = 0.67$
$\bar{y}_{6,3} = 0.67$
$\bar{y}_{9,4} = 0.50$
$\bar{y}_{5,4} = 0.50$

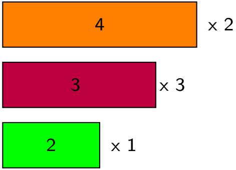


ONE-CUT is included in ARC-FLOW

We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

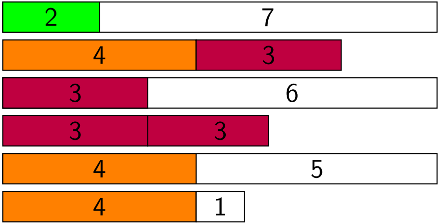
Set all $\bar{x}_{d,e} = 0$;
For all $\bar{y}_{p,q} > 0$ do
  $\bar{x}_{c-p,c-p+q} \mathrel{+}= \bar{y}_{p,q}$;
  $\bar{x}_{c-p+q,c} \mathrel{+}= \bar{y}_{p,q}$;
  if $p \neq c$
    $\bar{x}_{c-p,c} \mathrel{-}= \bar{y}_{p,q}$;
  endif
end do;

Example (c=9):

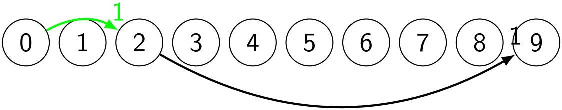


LP solution for ONE-CUT ($z = 2.17$):

$\bar{y}_{9,2} = 1.00$
$\bar{y}_{7,4} = 1.00$
$\bar{y}_{9,3} = 0.67$
$\bar{y}_{6,3} = 0.67$
$\bar{y}_{9,4} = 0.50$
$\bar{y}_{5,4} = 0.50$



Resulting LP solution for ARC-FLOW:

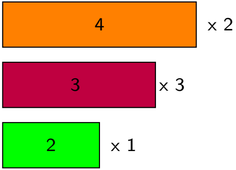


ONE-CUT is included in ARC-FLOW

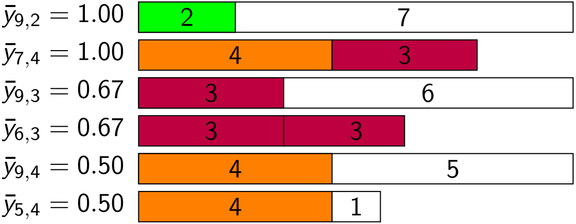
We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

Set all $\bar{x}_{d,e} = 0$;
For all $\bar{y}_{p,q} > 0$ do
  $\bar{x}_{c-p,c-p+q} \mathrel{+}= \bar{y}_{p,q}$;
  $\bar{x}_{c-p+q,c} \mathrel{+}= \bar{y}_{p,q}$;
  if $p \neq c$
    $\bar{x}_{c-p,c} \mathrel{-}= \bar{y}_{p,q}$;
  endif
end do;

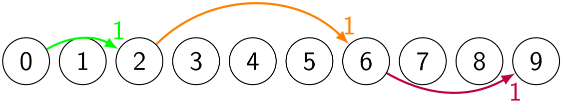
Example (c=9):



LP solution for ONE-CUT ($z = 2.17$):



Resulting LP solution for ARC-FLOW:

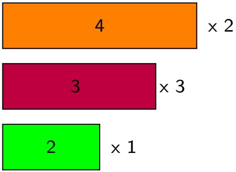


ONE-CUT is included in ARC-FLOW

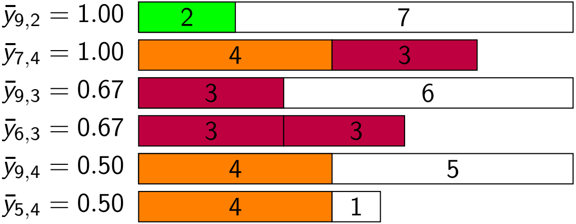
We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

Set all $\bar{x}_{d,e} = 0$;
For all $\bar{y}_{p,q} > 0$ do
  $\bar{x}_{c-p,c-p+q} \mathrel{+}= \bar{y}_{p,q}$;
  $\bar{x}_{c-p+q,c} \mathrel{+}= \bar{y}_{p,q}$;
  if $p \neq c$
    $\bar{x}_{c-p,c} \mathrel{-}= \bar{y}_{p,q}$;
  endif
end do;

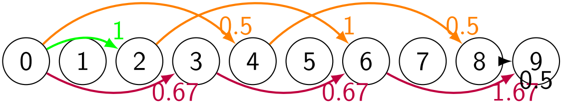
Example (c=9):



LP solution for ONE-CUT ($z = 2.17$):



Resulting LP solution for ARC-FLOW:



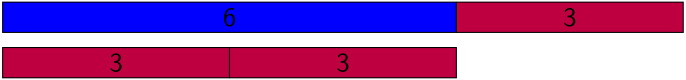
ONE-CUT is included in ARC-FLOW

We use an LP ONE-CUT solution $[\bar{y}_{pq}]$ to build an LP ARC-FLOW solution $[\bar{x}_{de}]$

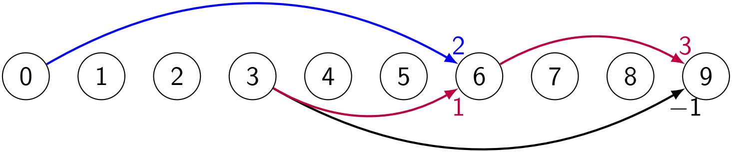
ONE-CUT does not forbid recutting an item

$\bar{y}_{9.6} = 2$

$\bar{y}_{6,3} = 1$



Which would result into a negative flow in ARC-FLOW



We solve this by using an equivalent Normal ONE-CUT formulation where we add

$$\sum_{p \in B(q)} y_{p+q,p} \geq \sum_{r \in C(q)} y_{qr} \quad q \in W$$

Improving ARC-FLOW with a Multi-Graph

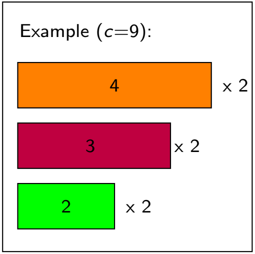
Use of a multi-graph to possibly increase arcs length

- *Brandão and Pedroso (Computers & Operations Research, 2016)*
- $x_{de}^{j} = \text{n}^{\circ}$ of times item j is packed as an arc (d, e)
- enlarge width of each arc (d, e, j) by solving a subset sum problem
- remove arcs (d, e, j) if there is another arc (d', e', j) s.t. $d' \geq d$, $e' \leq e$.

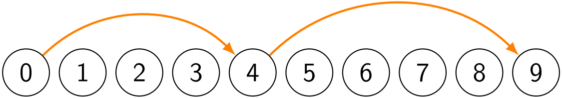
Improving ARC-FLOW with a Multi-Graph

Use of a multi-graph to possibly increase arcs length

- *Brandão and Pedroso (Computers & Operations Research, 2016)*
- x^j_{de} = n° of times item j is packed as an arc (d, e)
- enlarge width of each arc (d, e, j) by solving a subset sum problem
- remove arcs (d, e, j) if there is another arc (d', e', j) s.t. $d' \geq d$, $e' \leq e$.



Multi-graph

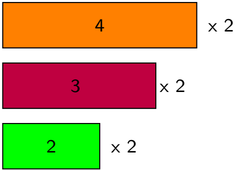


Improving ARC-FLOW with a Multi-Graph

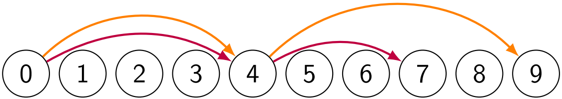
Use of a multi-graph to possibly increase arcs length

- *Brandão and Pedroso (Computers & Operations Research, 2016)*
- x^j_{de} = n° of times item j is packed as an arc (d, e)
- enlarge width of each arc (d, e, j) by solving a subset sum problem
- remove arcs (d, e, j) if there is another arc (d', e', j) s.t. $d' \geq d$, $e' \leq e$.

Example (c=9):



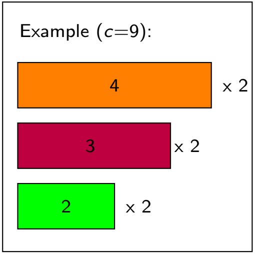
Multi-graph



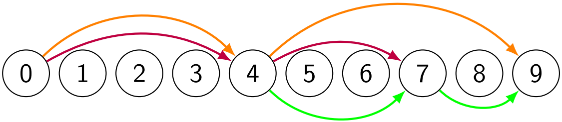
Improving ARC-FLOW with a Multi-Graph

Use of a multi-graph to possibly increase arcs length

- *Brandão and Pedroso (Computers & Operations Research, 2016)*
- x_{de}^{j} = n° of times item j is packed as an arc (d, e)
- enlarge width of each arc (d, e, j) by solving a subset sum problem
- remove arcs (d, e, j) if there is another arc (d', e', j) s.t. $d' \geq d$, $e' \leq e$.



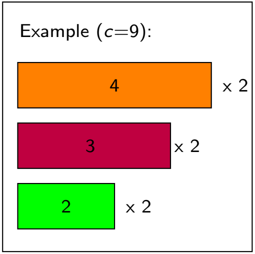
Multi-graph



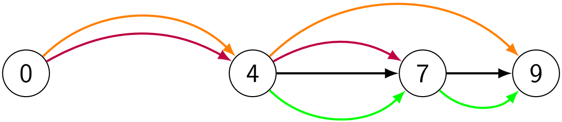
Improving ARC-FLOW with a Multi-Graph

Use of a multi-graph to possibly increase arcs length

- *Brandão and Pedroso (Computers & Operations Research, 2016)*
- x^j_{de} = n$^\circ$ of times item j is packed as an arc (d, e)
- enlarge width of each arc (d, e, j) by solving a subset sum problem
- remove arcs (d, e, j) if there is another arc (d', e', j) s.t. $d' \geq d$, $e' \leq e$.



Multi-graph

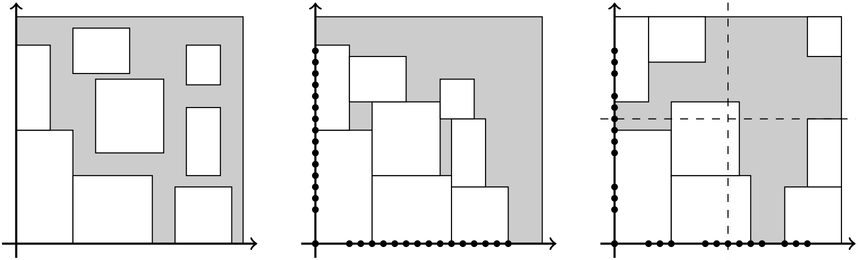


8 arcs (instead of the original 27)

Improving ARC-FLOW with Meet-in-the-Middle Patterns

Proposed in *Côté and Iori, The Meet-in-the-Middle Principle for C&P Problems (2016, submitted)* for general C&P problems

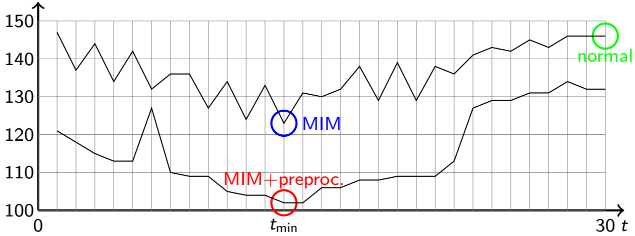
- *normal patterns* align items to the left to decrease candidate optimal solutions
- proposed by *Herz (IBM J. of Res. and Dev., 1972)*, *Christofides and Whitlock (OR, 1977)*, improved by *Boschetti et al. (IMA J. of Man. Math., 2002)*
- *Meet-in-the-Middle* (MIM) patterns align items either to the left or to the right according to a threshold cut value t



Improving ARC-FLOW with Meet-in-the-Middle Patterns

Key note features of the MIM patterns:

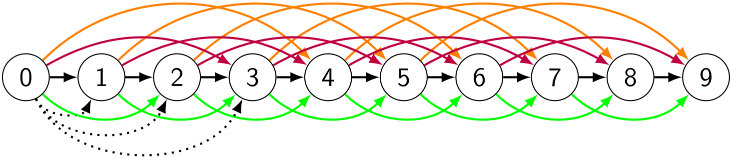
- computed by easy DP that also finds the best threshold cut value t
- same time complexity than normal patterns
- size is never higher and usually much smaller
- further reduction criteria
- when used in ARC-FLOW for CSP, 50% average reduction in the number of arcs and 97% max reduction



The case of Variable-Sized (and Cost) BPP (VSBPP)

VSBPP is the BPP variant in which each bin $k \in K$ has capacity c_k and cost γ_k

ARC-FLOW is easily adapted by including new arcs $(0, c_{\max} - c_k)$ for each $k \in K$ (slight variation of *Valério de Carvalho (EJOR, 2002)*)



Proposition

ONE-CUT(VSBPP) is equivalent to ARC-FLOW(VSBPP)

Improvements based on Multi-graph and MIM can be directly used

Preliminary computational results

Number of VSBPP/BPP instances solved in 1 hour (3.10GHz, 4 cores, Cplex 12.6)

			ARC-FLOW			Multi-Graph			MIM		
Instance	c_{max}	#inst	#opt	sec	#var	#opt	sec	Δvar	#opt	sec	Δvar
Monaci	150	300	**300**	1	1 K	**300**	0	-49%	**300**	1	-48%
Perboli 2	300	60	**60**	1	5 K	**60**	1	-5%	**60**	1	-5%
Perboli 3	120	480	**480**	0	1 K	**480**	0	-20%	**480**	0	-20%
Belov 1	10 000	50	**50**	87	146 K	**50**	57	-25%	**50**	58	-46%
Belov 2	10 000	50	**50**	68	139 K	**50**	53	-29%	**50**	37	-50%
AI 200	2 500	50	49	233	121 K	**50**	45	-49%	**50**	110	-30%
AI 400	10 000	50	3	3 480	940 K	10	3 333	-45%	10	3 123	-29%
ANI 200	2 500	50	27	2 064	119 K	**50**	30	-48%	32	1 510	-30%
ANI 400	10 000	50	0	3 602	935 K	5	3 465	-87%	2	3 536	-29%
Sum/Avg		1140	1019	1060	267 K	1055	776	-41%	1034	931	-35%

Don't blame capacity

Pseudo-polynomial formulations are interesting both in theory and in practice

Of course, it is inadvisable to use them if capacity is too large

Studying combinations with B&P is of great interest

MILP solver matters

Preliminary results on the $P||\sum w_j C_j$, by *Arthur Kramer, Dell'Amico, Iori (2016, work in progress)*

- Selected instances solved with ARC-FLOW and 300 seconds (at 2.4GHz)

LP method	Solver	#inst	#opt	#nodes	seconds
Dual Simplex	Gurobi 6.5.1	260	**212**	0.2	81.6
	Cplex 12.6.2	260	195	0.5	90.6
Primal Simplex	Gurobi 6.5.1	260	**250**	5.6	38.3
	Cplex 12.6.2	260	224	8.9	58.6
Barrier	Gurobi 6.5.1	260	**258**	13.2	26.9
	Cplex 12.6.2	260	225	22.0	54.5

Rectangle packing

In two-dimensional orthogonal C&P problems, good computational results have been obtained by using pseudo-polynomial models in primal decomposition methods:

- *Côté, Dell'Amico, Iori, Combinatorial Benders' Cuts for the Strip Packing Problem, (Operations Research, 2014)*
- *Delorme, Iori, Martello, Logic Based Benders' Decomposition for Orthogonal Stock Cutting Problems, (2016, submitted)*

Use B&P as one of the engines in this kind of approaches? (Several instances with 20 items are open problems since decades)

The MIRUP conjecture

MIRUP conjecture holds for both BPP and CSP and states that the difference between z_{opt} and the optimal rounded-up GG solution value is at most 1

- *Caprara, Dell'Amico, Díaz Díaz, Iori, Rizzi, Friendly Bin Packing Instances without Integer Round-up Property, Mathematical Programming (2015)*
- *Kartak, Ripatti, Scheithauer, Kurz, Minimal proper non-IRUP instances of the one-dimensional cutting stock problem, Discrete Applied Mathematics (2015)*

Can pseudo-polynomial formulations help in this context?

Thank you very much for your attention