Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms

Maxence Delorme[(1)], Manuel Iori[(2)], Silvano Martello[(1)]

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

BOLOGNA 2015

Outline

The Bin Packing Problem (BPP)

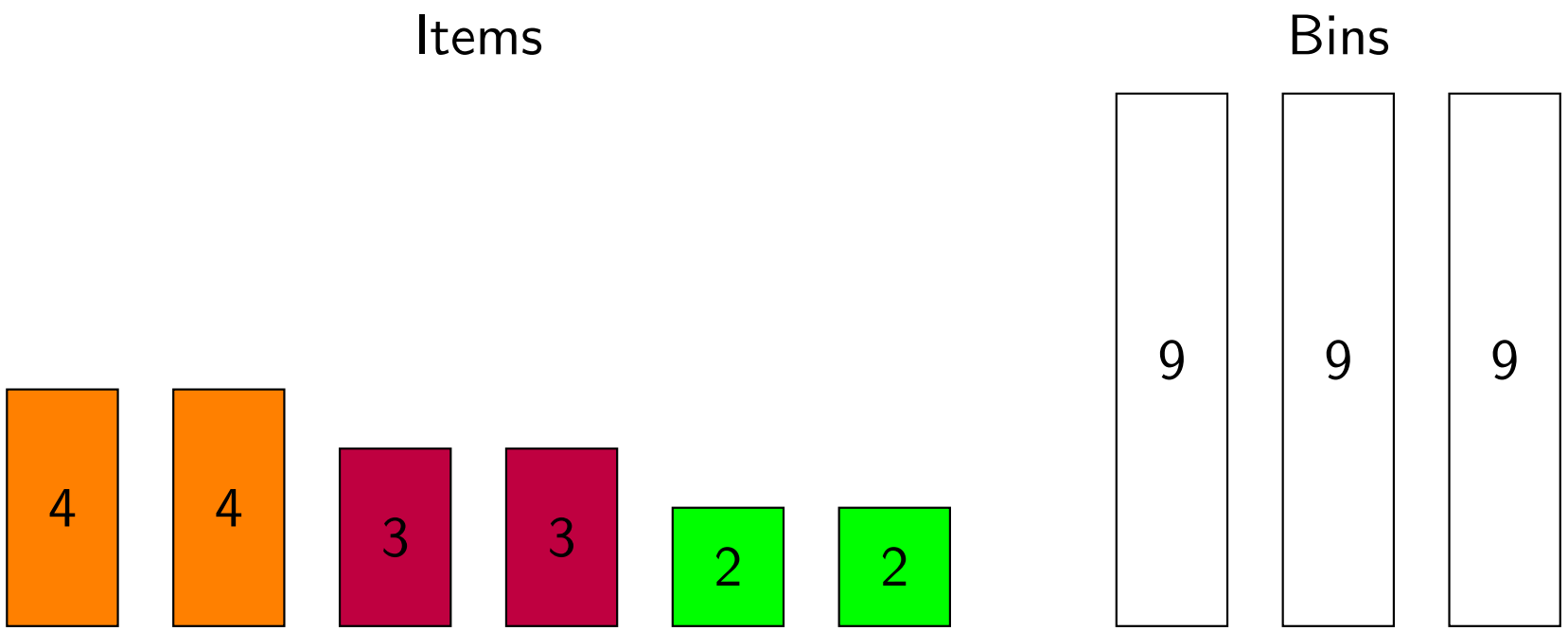
Classical Bin Packing Problem

Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.

The Bin Packing Problem (BPP)

Classical Bin Packing Problem

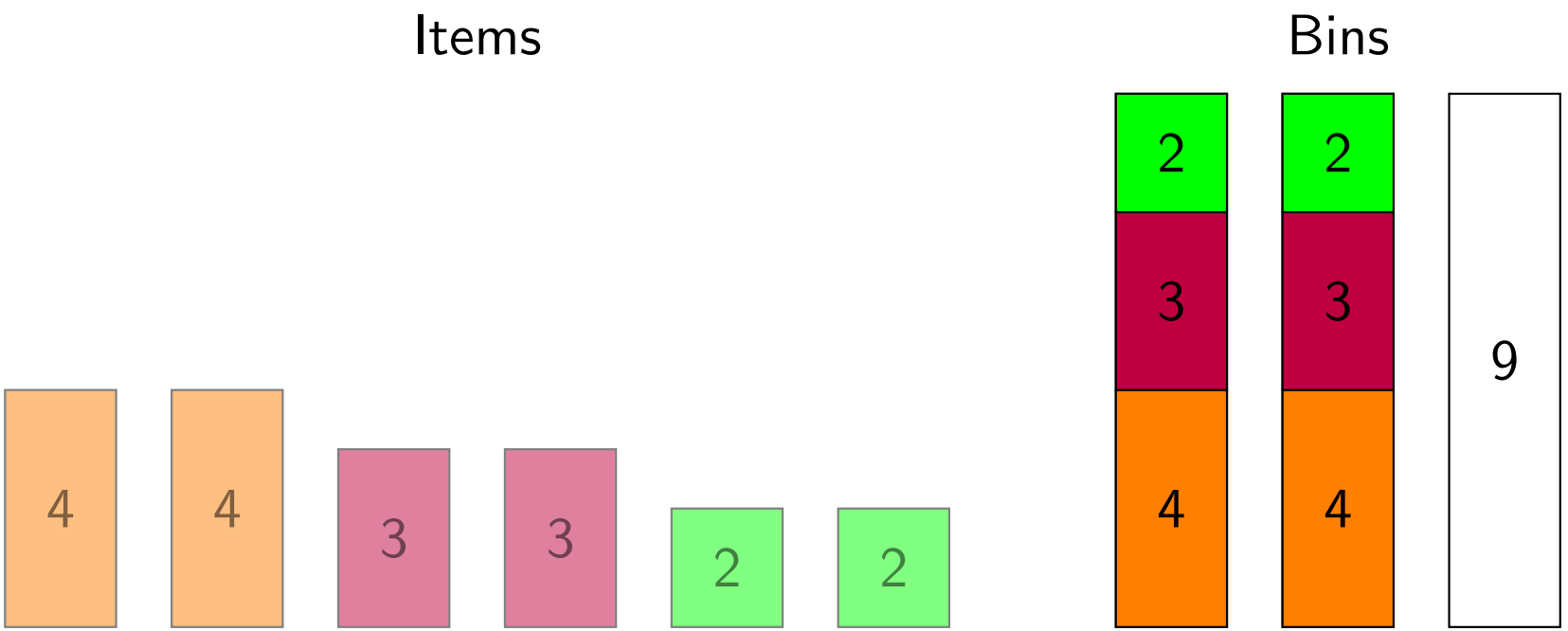
Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.



The Bin Packing Problem (BPP)

Classical Bin Packing Problem

Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.



The Cutting Stock Problem (CSP)

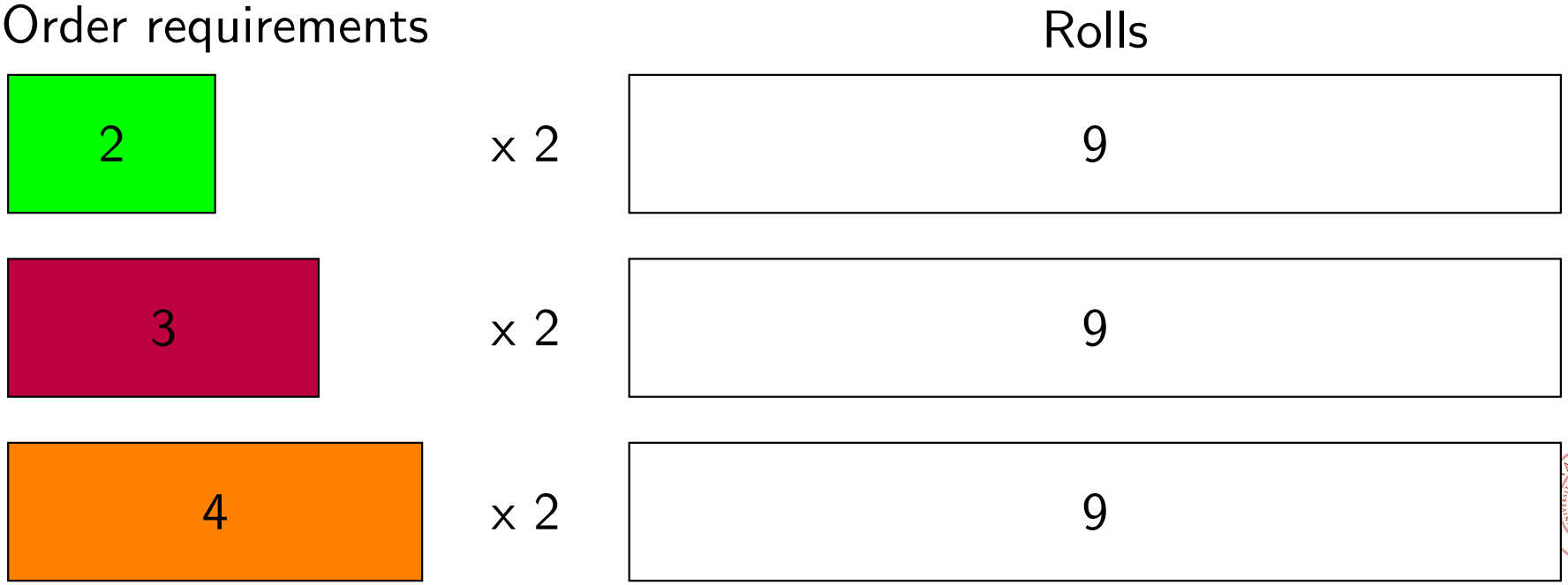
Classical Cutting Stock Problem

Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.

The Cutting Stock Problem (CSP)

Classical Cutting Stock Problem

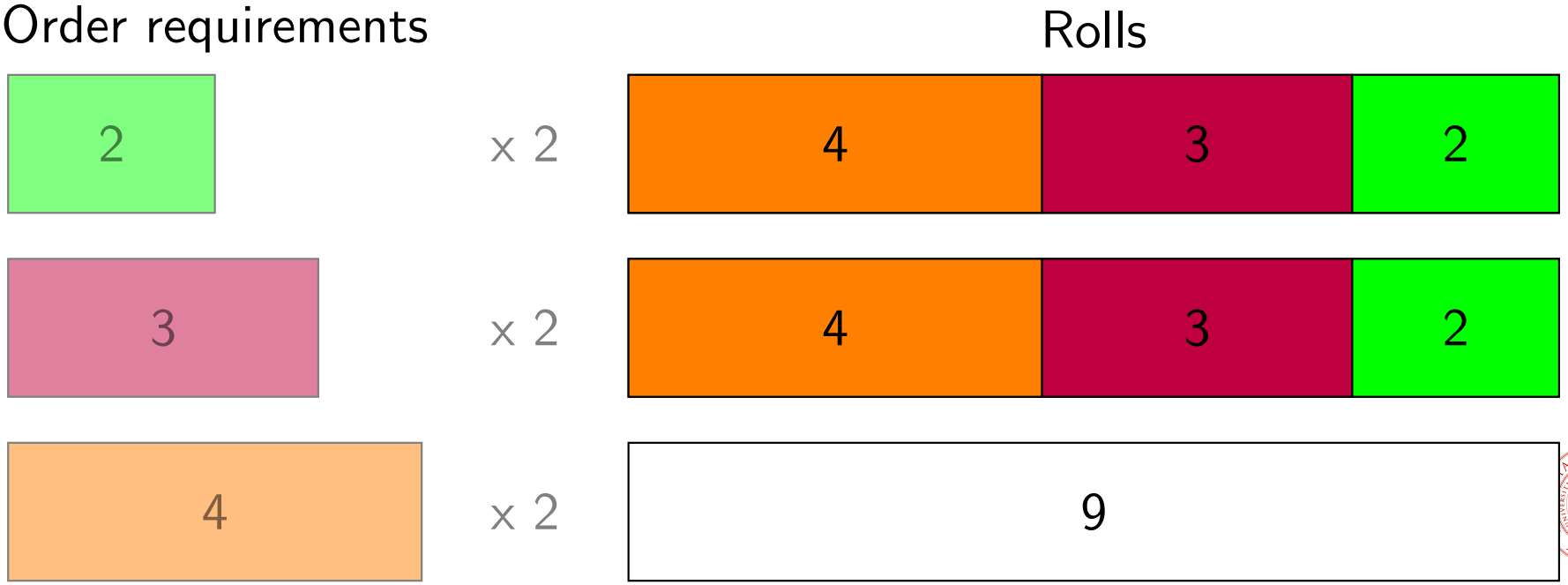
Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.



The Cutting Stock Problem (CSP)

Classical Cutting Stock Problem

Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.

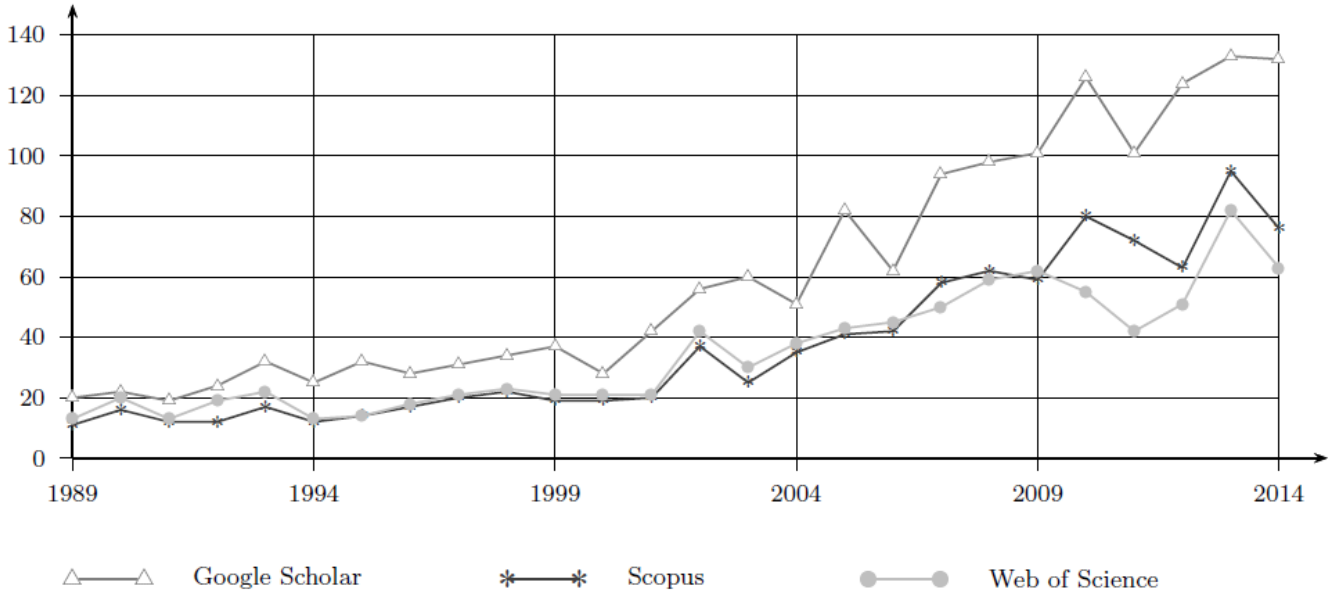


Some applications for the BPP and the CSP

- Cutting materials in industry (wood, paper, aluminium ...)
  - Kantorovich, Mathematical methods of organizing and planning production, *Management Science*, 1960 (originally from 1939)
  - Stadler, A one-dimensional cutting stock problem in the aluminium industry and its solution, *European Journal of Operational Research*, 1990
- Loading when 1 dimension is considered (file storage on computers, container loading ...)
- Solving more difficult problems (vehicle routing, multi-dimensional BPP, ...)

Context of our work

- BPP and CSP are more and more studied by researchers
  The subject is of interest
- Many techniques can be used to solve the BPP and the CSP
  A survey is relevant
- All the literature instances are solved
  A will to create a new computational challenge



Objectives of our work

- Gather in a survey the most important articles
- Test the efficiency of some exact methods proposed in the literature (Branch-and-Bound, Branch-and-Price, Pseudo-Polynomial models ...)
- Study the behaviour of those methods when the parameters of the test instances change
- Propose new instances that are difficult to solve in practice

The textbook BPP model solved by ILP solver

Martello, Toth, Knapsack Problems: Algorithms and Computer Implementations, *Wiley*, 1990
Roots in the seminal work by Kantorovich, Mathematical methods of organizing and planning production, *Management Science*, 1960 (originally from 1939)

$$\min \quad \sum_{i=1}^{m} y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_{ij} \leq c y_i \qquad (i = 1, \ldots, m), \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad (j = 1, \ldots, n), \tag{3}$$

$$y_i \in \{0, 1\} \qquad (i = 1, \ldots, m), \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m; j = 1, \ldots, n). \tag{5}$$

Easy to implement, may be efficient on small instances ($n \leq 100$)

Branch & Bound algorithms

Martello, Toth, Knapsack Problems: Algorithms and Computer Implementations, *Wiley*, 1990
Scholl, Klein, Jürgens, BISON: a fast hybrid procedure for exactly solving the one-dimensional BPP, *Computers & Operations Research*, 1997

Main idea: Explore (in a "smart" way) through an enumeration tree all the feasible packings
Smart because:

- Use of reduction procedures
- Use of a set of specially designed Lower Bounds (L1, L2, L3 ... L6)
- Use of fast and good heuristics to get Upper Bounds

Avoid the use of solvers. Generally good on small instances ($n \leq 100$), or when the preprocessing is efficient.

Branch & Price algorithms

Gilmore, Gomory, A linear programming approach to the CSP, *Operations research*, 1963
Vance, Barnhart, Johnson, Nemhauser, Solving binary CSP by column generation and branch-and-bound, *Computational optimization and applications*, 1994
Belov, Scheithauer, A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting, *European Journal of Operational Research*, 2006

Main idea: Consider the Bin Packing Problem as a Set Covering Problem

Set Covering Problem (SCP)

Given a set of elements $\{1, 2, ..., n\}$ (called the universe) and a family S of m sets whose union equals the universe, the set covering problem is to identify the smallest subfamily of S whose union equals the universe.

- The elements $\{1, 2, ..., n\}$ are the items
- The family S is composed by every feasible bin (column generation used to generate bins)

Very good when the lower bound is equal to the optimum (true in 99% of the case), conjectured to be at most one bin away from the optimum (called Non-IRUP).

Constraints Programming

Shaw, A constraint for bin packing, 2004
Schaus, Régin, Van Schaeren, Dullaert, Raa, Cardinality reasoning for bin-packing constraint: Application to a tank allocation problem, 2012

Main idea: Use properties of feasible solutions to define the search domain. Shaw did a very good job and created a constraint implemented in CPLEX.

Can be efficient on small instances ($n \leq 100$) and has the advantage of easily allowing additional constraints.

Pseudo Polynomial Models solved by ILP solver

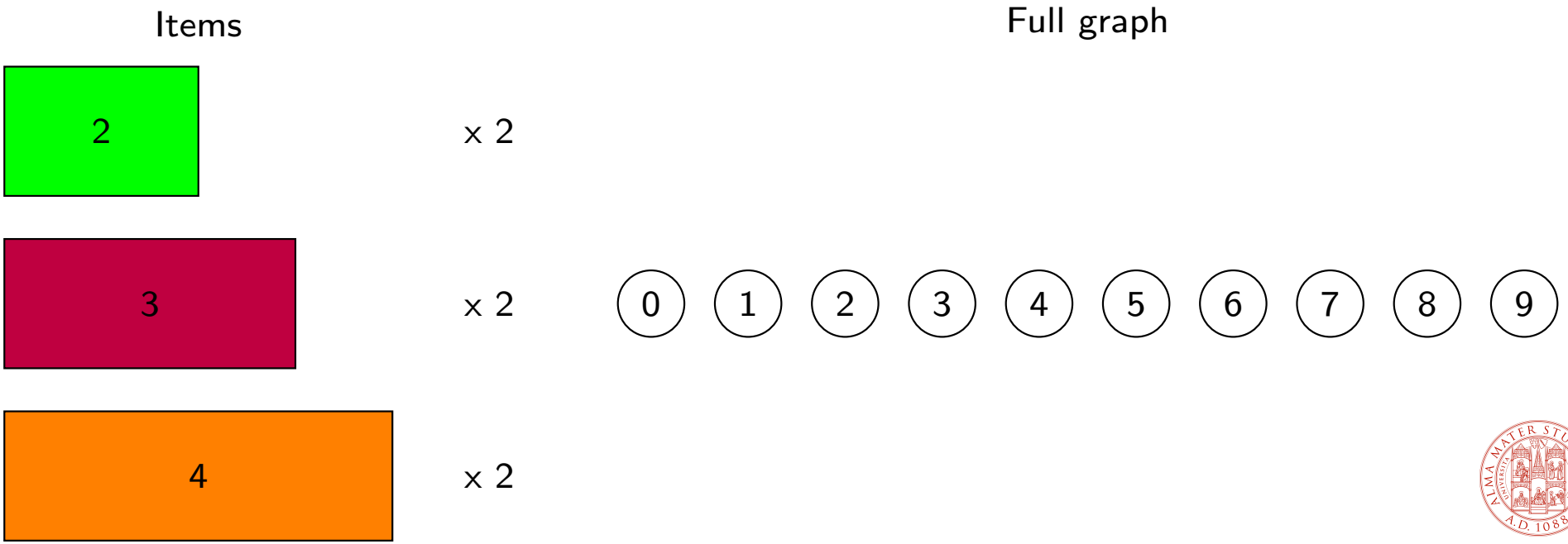
Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

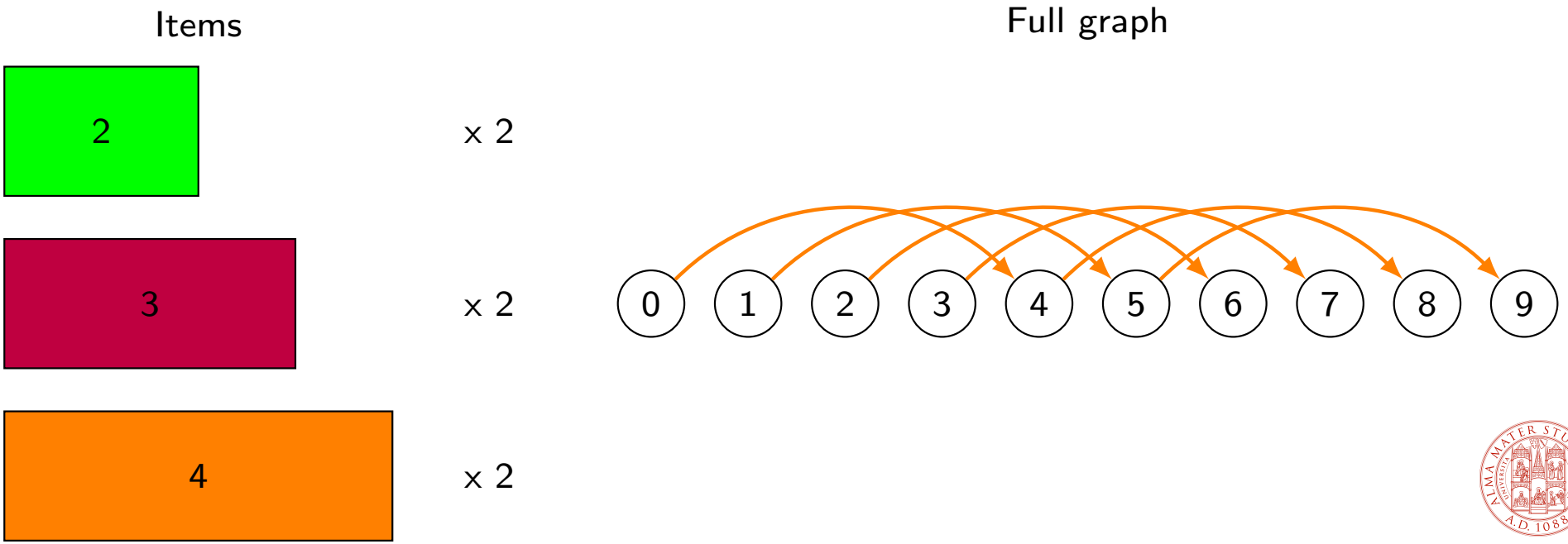
Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

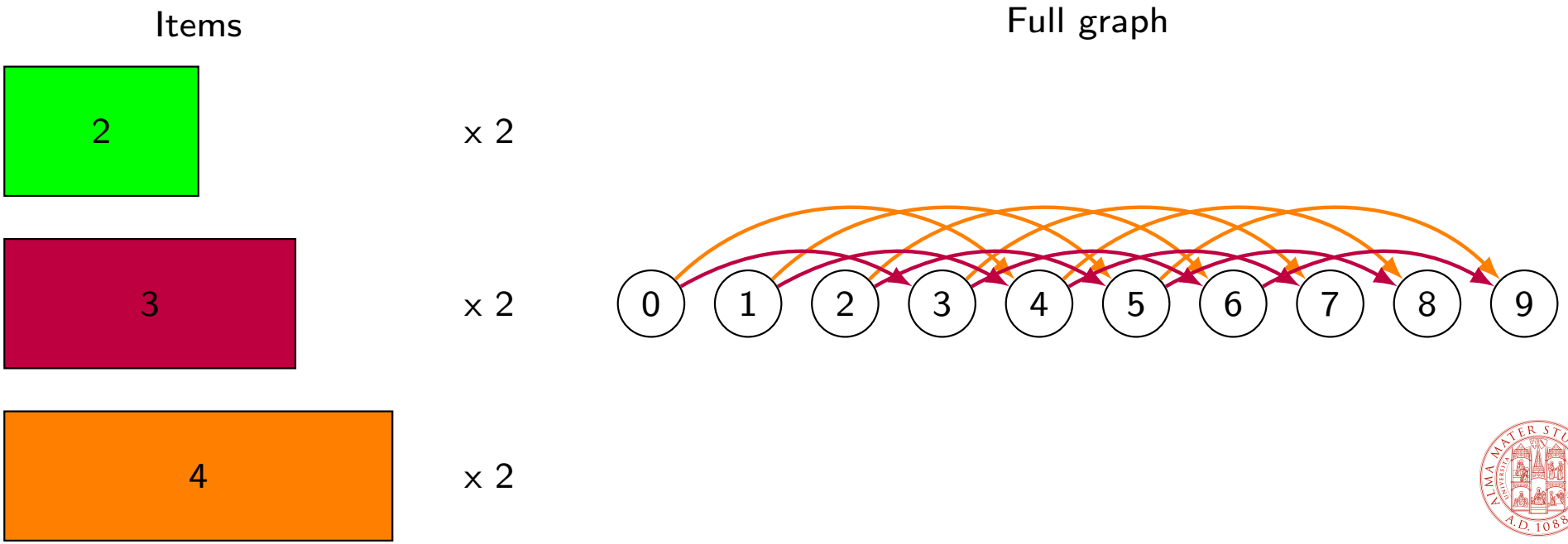
Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

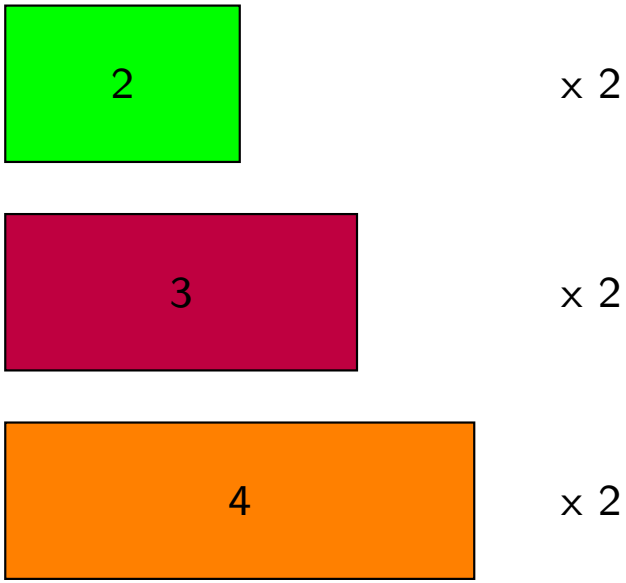


Pseudo Polynomial Models solved by ILP solver

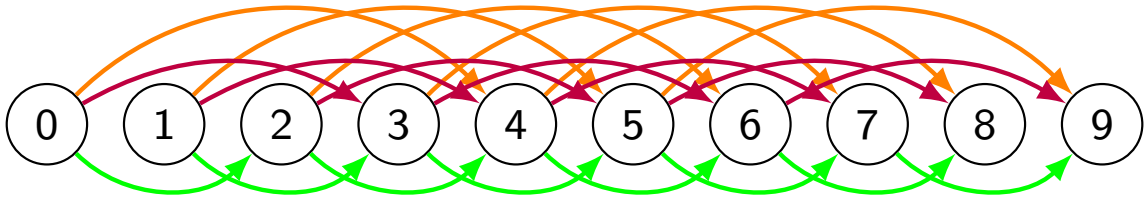
Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

Items



Full graph

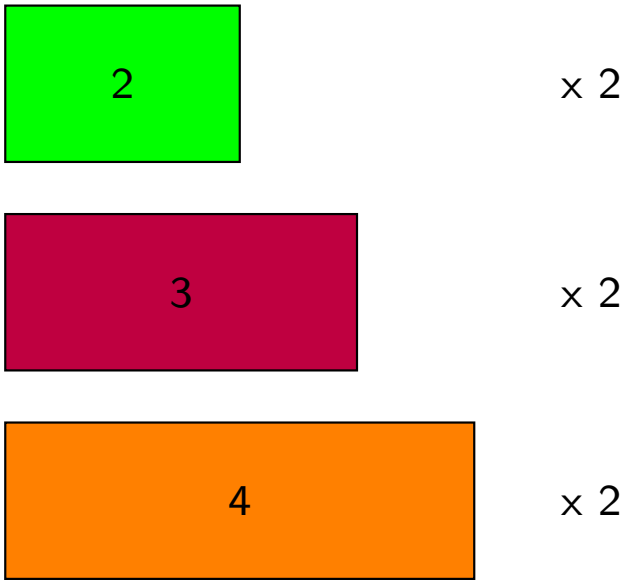


Pseudo Polynomial Models solved by ILP solver

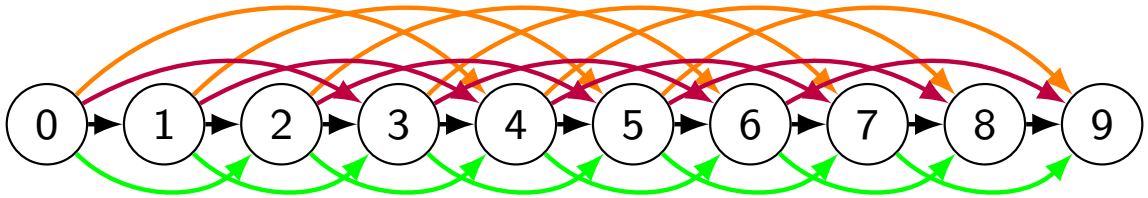
Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

Items



Full graph: 30 arcs, 10 nodes

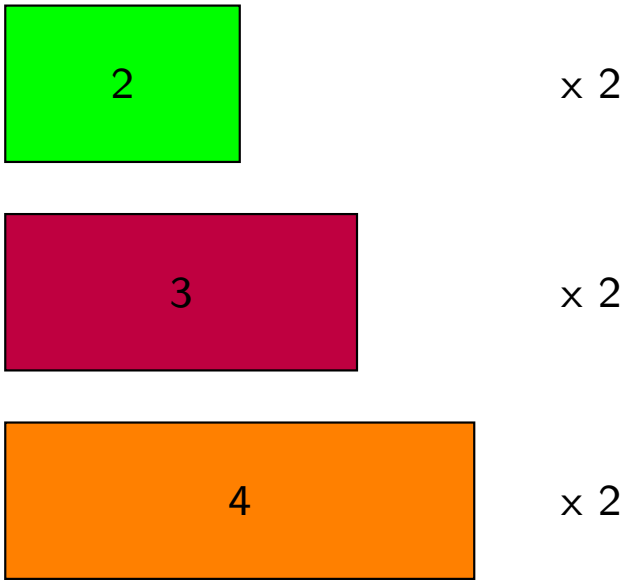


Pseudo Polynomial Models solved by ILP solver

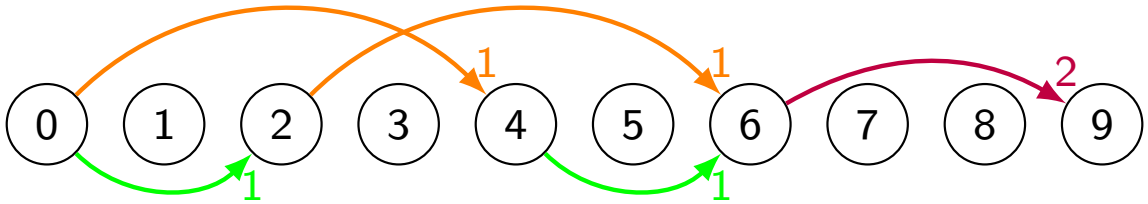
Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

Items



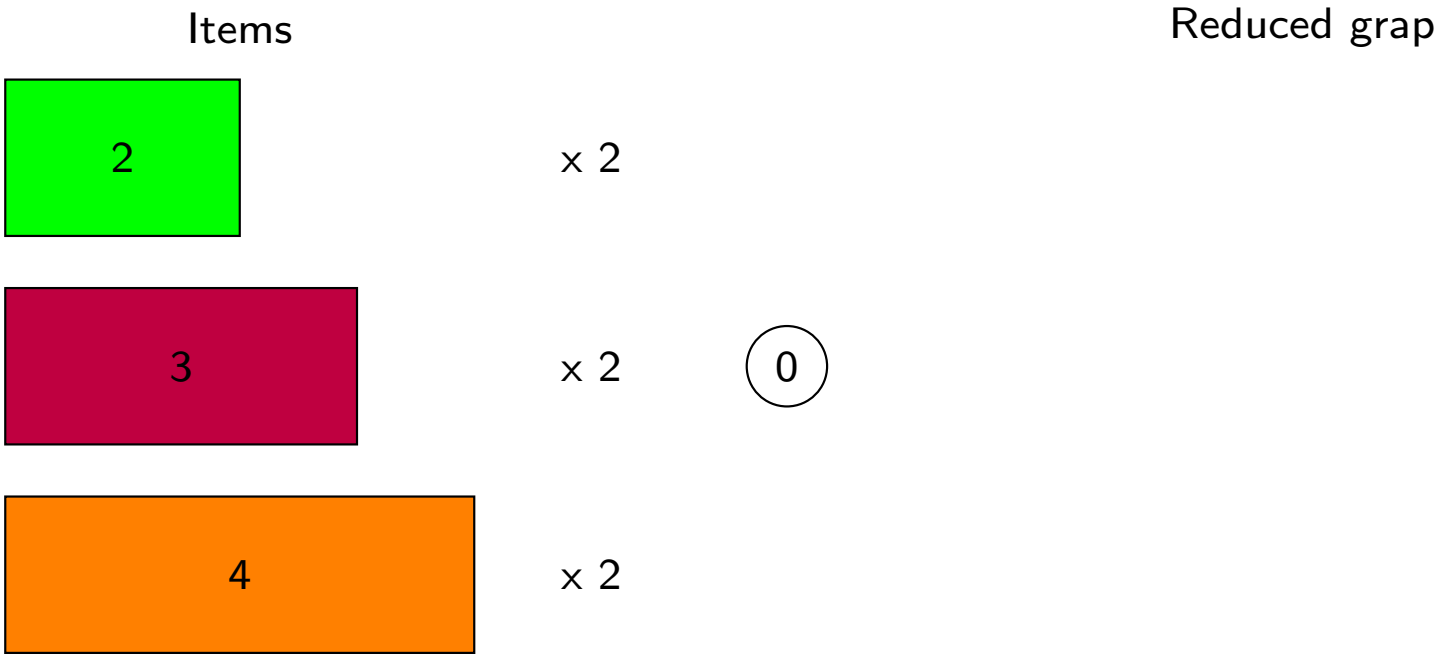
Optimal solution



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

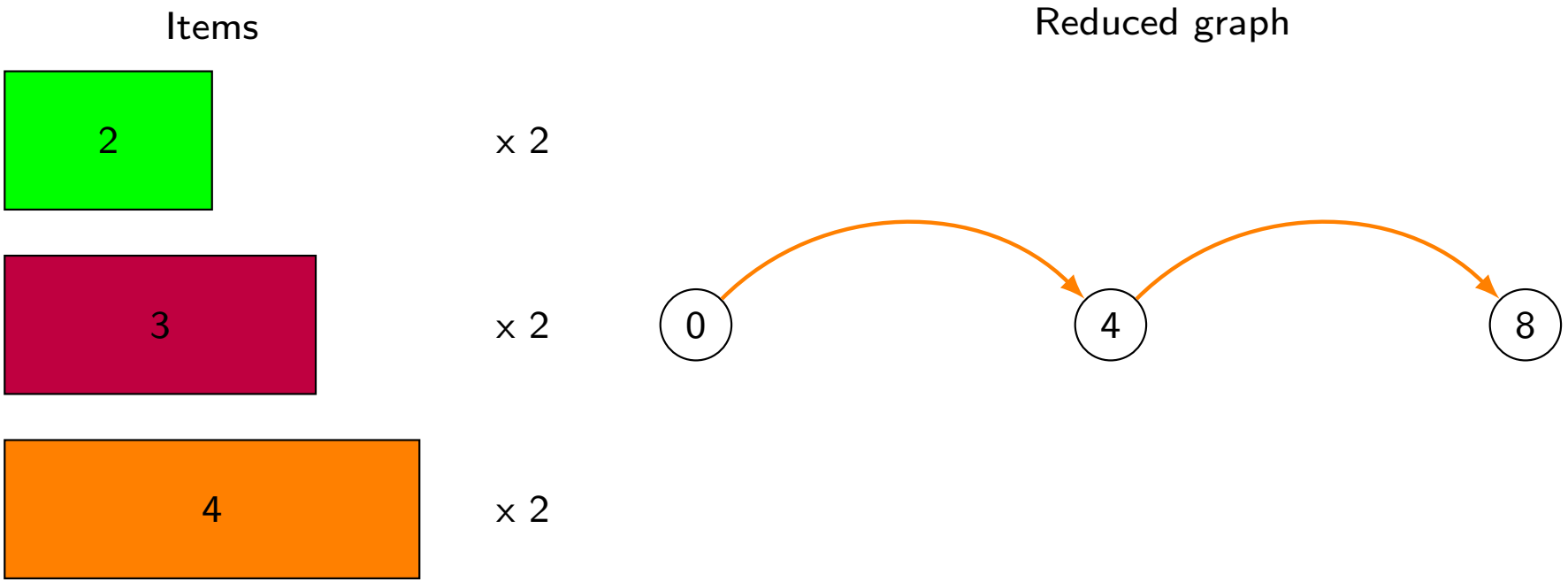
Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

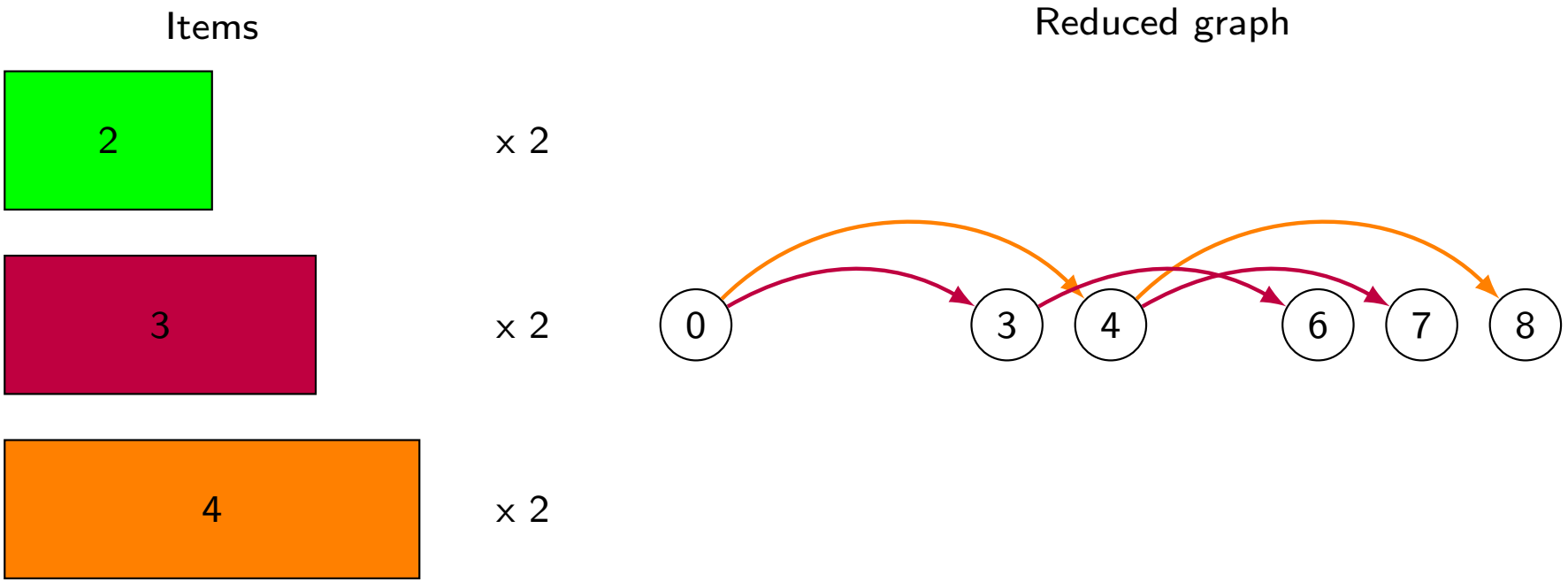
Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

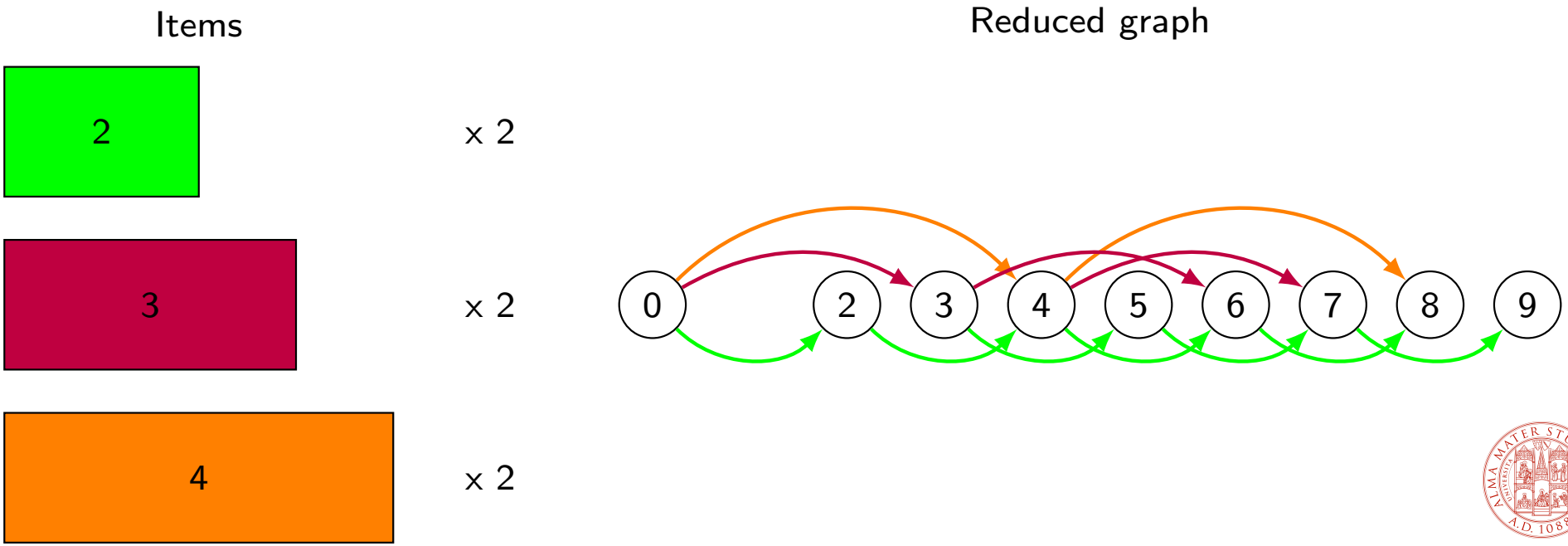
Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

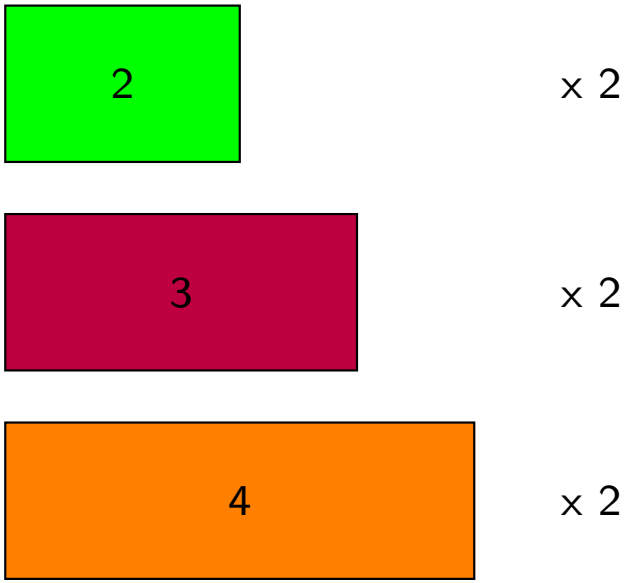


Pseudo Polynomial Models solved by ILP solver

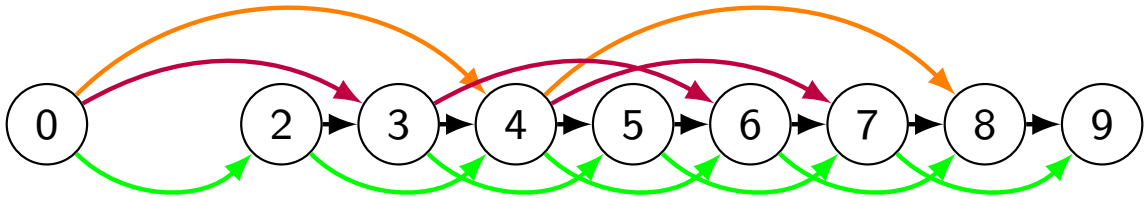
Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

Items



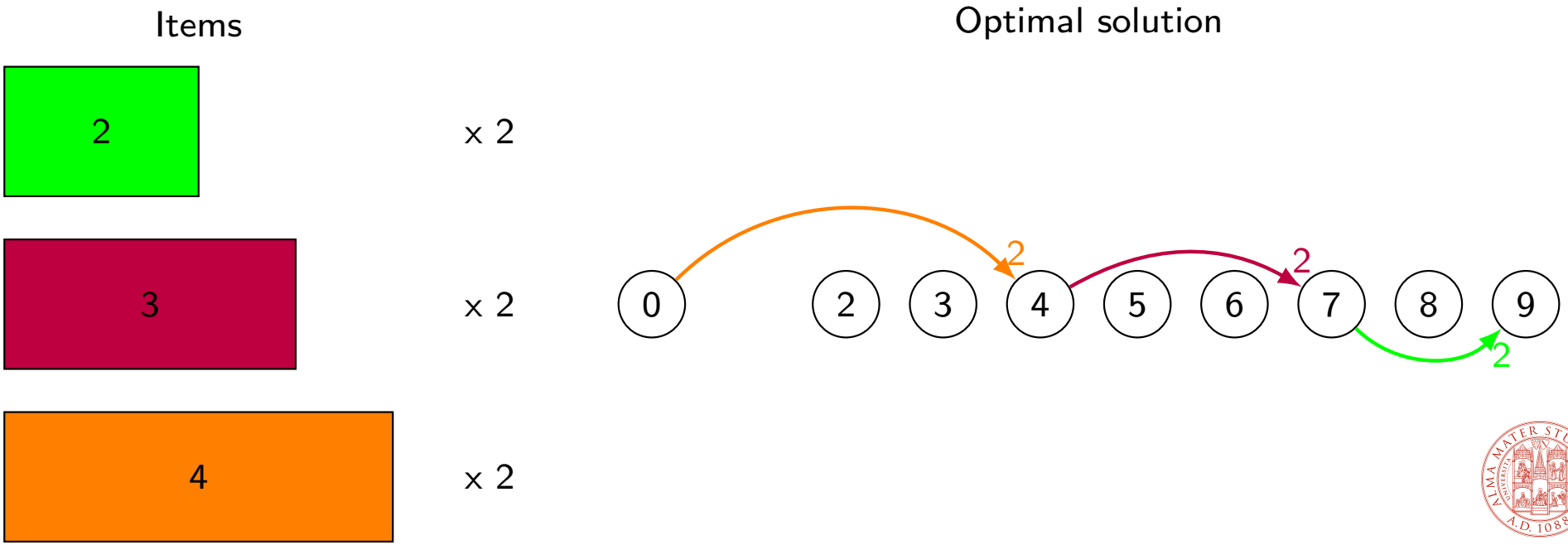
Reduced graph: 19 arcs, 9 nodes



Pseudo Polynomial Models solved by ILP solver

Dyckhoff. A new linear programming approach to the cutting stock problem, *Operations Research* 1981
Valerio de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* 1999
Cambazard, O'Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items



Pseudo Polynomial Models solved by ILP solver

Model

$$\min \quad z \tag{6}$$

$$\text{s.t.} \quad -\sum_{d\in\delta^-(e)} x_{de} + \sum_{f\in\delta^+(e)} x_{ef} = \begin{cases} z & \text{if } e=0; \\ -z & \text{for } e=c; \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

$$\sum_{(d,d+w_i)\in A'} x_{d,d+w_i} \geq b_i \qquad (i=1,...,m), \tag{8}$$

$$x_{de} \geq 0 \text{ and integer} \qquad (d,e)\in A', \tag{9}$$

where $\delta^-(e)$ (resp. $\delta^+(e)$) denotes the set of arcs entering (resp. emanating from) e.

Very good when the lower bound is equal to the optimum and/or when the number of arcs is not too big (c $<$ 1000) .

Instances

- Literature instances (1615)
  - 1210 instances from Sholl, Klein and Juergens (1997)
  - 28 instances from Schoenfield (2002)
  - 200 instances from Schwerin and Waescher (1998).
  - 17 instances from Waescher and Gau (1996).
  - 160 instances from Falkenauer (1996).
- Randomly generated instances (3840)
  - Variable number of items: 50,...,1000
  - Variable bin capacity: 50,...,1000
  - Variable parameters for the distribution of the items
  - 10 instances for each combination
- Augmented Non-IRUP (ANI) instances (250 + 250 Augmented IRUP (AI))
  - Variable number of items: 201, 402, 600, 801, and 1002
  - Each have its own maximum capacity: 2 500, 10 000, 20 000, 40 000, and 80 000
  - 50 instances for each of the 5 sets
  - An AI instance was created from each ANI instances by splitting 1 item so that the Non-IRUP is lost

Results for literature instances

Number of literature instances (average gap wrt lower bound) solved in less than one minute, on an Intel Xeon 3.10GHz with 8GB of RAM

Set	tested inst.	Branch-and-bound			Branch-and-price			Pseudo-polynomial				Others	
		MTP	BISON	CVRPSEP	VANCE	BELOV	SCIP-BP	ONECUT	ARCFLOW	DPFLOW	VPSOLVER	BASIC ILP	CSTRPROG
Falkenauer U	74	22 (1.7)	44 (0.4)	22 (1.8)	53 (1.2)	**74** (0.0)	18 (2.1)	**74** (0.0)	**74** (0.0)	37 (1.8)	**74** (0.0)	10 (2.3)	28 (2.0)
Falkenauer T	80	6 (7.0)	42 (0.5)	0 (11.0)	76 (0.1)	57 (0.3)	35 (4.5)	**80** (0.0)	**80** (0.0)	40 (8.8)	**80** (0.0)	7 (7.0)	39 (8.8)
Scholl 1	323	242 (0.3)	288 (0.1)	223 (0.3)	**323** (0.0)	**323** (0.0)	244 (0.2)	**323** (0.0)	**323** (0.0)	289 (0.1)	**323** (0.0)	212 (0.3)	90 (0.6)
Scholl 2	244	130 (0.6)	233 (0.0)	65 (1.4)	204 (0.2)	**244** (0.0)	67 (1.2)	118 (0.4)	202 (0.1)	58 (1.3)	208 (0.1)	90 (1.0)	122 (1.3)
Scholl 3	10	0 (1.5)	3 (0.7)	0 (4.1)	**10** (0.0)	**10** (0.0)	0 (4.1)	0 (4.1)	0 (4.1)	0 (4.1)	**10** (0.0)	0 (2.7)	0 (4.1)
Wäscher	17	0 (1.0)	10 (0.4)	0 (1.0)	6 (0.6)	**17** (0.0)	0 (1.0)	0 (1.0)	0 (1.0)	0 (1.0)	6 (0.6)	4 (0.8)	7 (0.6)
Schwerin 1	100	15 (0.9)	**100** (0.0)	9 (0.9)	**100** (0.0)	**100** (0.0)	0 (1.0)	**100** (0.0)	**100** (0.0)	0 (1.0)	**100** (0.0)	32 (0.7)	**100** (0.0)
Schwerin 2	100	4 (1.4)	63 (0.4)	0 (1.4)	**100** (0.0)	**100** (0.0)	0 (1.4)	**100** (0.0)	**100** (0.0)	0 (1.4)	**100** (0.0)	36 (0.7)	60 (0.8)
Hard28	28	0 (1.0)	0 (1.0)	0 (1.0)	11 (0.6)	**28** (0.0)	7 (0.8)	6 (0.8)	16 (0.4)	0 (1.0)	27 (0.0)	0 (1.0)	0 (1.0)
Total	976	419 (0.9)	783 (0.1)	319 (1.4)	883 (0.1)	953 (0.0)	371 (1.0)	801 (0.2)	895 (0.1)	424 (1.2)	928 (0.0)	391 (1.0)	446 (1.3)

- Best methods seem to be the B&P algorithm of Belov and the Pseudo Polynomial models
- All instances can be solved in less than 10 minutes by BELOV

Results for randomly generated instances

Number of random instances solved in less than one minute (average gap wrt lower bound) when varying n, on an Intel Xeon 3.10GHz with 8GB of RAM

n	tested inst.	Branch-and-bound			Branch-and-price			Pseudo-polynomial				Others	
		MTP	BISON	CVRPSEP	VANCE	BELOV	SCIP-BP	ONECUT	ARCFLOW	DPFLOW	VPSOLVER	BASIC ILP	CSTRPROG
50	165	163 (0.0)	**165** (0.0)	164 (0.0)	**165** (0.0)	**165** (0.0)	**165** (0.0)	**165** (0.0)	**165** (0.0)	**165** (0.0)	**165** (0.0)	157 (0.0)	71 (0.7)
100	271	243 (0.1)	257 (0.1)	239 (0.1)	**271** (0.0)	**271** (0.0)	**271** (0.0)	**271** (0.0)	**271** (0.0)	**271** (0.0)	**271** (0.0)	237 (0.1)	132 (0.6)
200	359	237 (0.4)	290 (0.2)	220 (0.6)	358 (0.0)	**359** (0.0)	293 (0.2)	358 (0.0)	**359** (0.0)	292 (0.2)	**359** (0.0)	201 (0.4)	171 (0.8)
300	393	166 (0.8)	265 (0.3)	144 (1.1)	387 (0.0)	**393** (0.0)	155 (0.8)	385 (0.0)	391 (0.0)	243 (0.6)	**393** (0.0)	115 (0.8)	140 (1.2)
400	425	151 (1.1)	244 (0.5)	138 (1.4)	416 (0.0)	**425** (0.0)	114 (1.1)	408 (0.1)	421 (0.0)	193 (1.1)	**425** (0.0)	92 (1.0)	104 (1.7)
500	414	121 (1.4)	208 (0.6)	128 (1.6)	394 (0.0)	**414** (0.0)	69 (1.7)	394 (0.1)	402 (0.0)	169 (1.3)	413 (0.0)	60 (1.5)	61 (2.0)
750	433	93 (2.0)	214 (0.7)	98 (2.3)	99 (2.1)	**433** (0.0)	22 (2.7)	401 (0.2)	415 (0.1)	120 (2.0)	431 (0.0)	54 (2.5)	23 (2.8)
1000	441	78 (2.6)	196 (0.8)	73 (3.1)	62 (2.8)	**441** (0.0)	0 (3.6)	407 (0.2)	416 (0.1)	67 (3.1)	434 (0.0)	39 (3.3)	7 (3.6)
Overall	2901	1252 (1.2)	1839 (0.5)	1204 (1.5)	2152 (0.8)	**2901** (0.0)	1089 (1.5)	2789 (0.1)	2840 (0.0)	1520 (1.2)	2891 (0.0)	955 (1.4)	709 (1.9)

- No change in the efficiency of the methods
- No difficulty for solving random instances even with "large" capacity and number of items

Results for ANI and AI instances

Number of difficult instances (ANI) solved in less than 1 hour (average gap wrt lower bound), on an Intel Xeon 3.10GHz with 8GB of RAM

n(ANI)	n(AI)	$\overline{c}$	ARCFLOW		BISON		BELOV		VPSOLVER	
			ANI	*AI*	ANI	*AI*	ANI	*AI*	ANI	*AI*
201	202	2500	16 (0.7)	*44 (0.1)*	0 (1.0)	*3 (0.9)*	**50** (0.0)	***50*** *(0.0)*	47 (0.1)	***50*** *(0.0)*
402	403	10000	0 (1.0)	*0 (1.0)*	0 (1.0)	*0 (1.0)*	1 (1.0)	*45 (0.1)*	6 (0.9)	*42 (0.2)*
600	601	20000	-	-	-	-	0 (1.0)	*21 (0.6)*	0 (1.0)	*8 (0.8)*
801	802	40000	-	-	-	-	0 (1.0)	*0 (1.0)*	0 (1.0)	*0 (1.0)*
1002	1003	80000	-	-	-	-	-	-	-	-
	Overall		16 (0.8)	*44 (0.6)*	0 (1.0)	*3 (1.0)*	51 (0.7)	*116 (0.4)*	53 (0.7)	*100 (0.5)*

- No change in the efficiency of the methods
- ANI instance are indeed hard to solve

Results for a subset of instances

Number of selected instances solved [average time in seconds] using different version of CPLEX, on an Intel Xeon 2.66GHz with 24 GB of RAM

Time	tested inst.	6.0 (1998)	7.0 (1999)	8.0 (2002)	9.0 (2003)	10.0 (2006)	11.0 (2007)	12.1 (2009)	12.6.0 (2013)
10 minutes	20	13 [366]	10 [420]	5 [570]	17 [268]	19 [162]	**20** [65]	19 [117]	**20** [114]
60 minutes	20	16 [897]	15 [1210]	15 [2009]	**20** [343]	**20** [186]	**20** [65]	19 [267]	**20** [114]

- Efficiency of pseudo-polynomial models highly depend on the performance of the ILP solver used

Conclusion

Conclusion

- According to our results, the best methods to solve the BPP and the CSP seem to be ARCFLOW, VPSOLVER and the B&P algorithm of Belov et al.
- The use of pseudo-polynomial models for solving the BPP became relevant thanks to the performance of the ILP solver
- There are some properties that make an instance difficult for the tool we tested
- Some new BPP instances are “open” !