Optim Lett (2018) 12:235–250
https://doi.org/10.1007/s11590-017-1192-z

ORIGINAL PAPER

BPPLIB: a library for bin packing and cutting stock problems

Maxence Delorme[1] · **Manuel Iori**[2] · **Silvano Martello**[1]

Received: 31 May 2017 / Accepted: 29 August 2017 / Published online: 4 September 2017

Abstract The bin packing problem (and its variant, the cutting stock problem) is among the most intensively studied combinatorial optimization problems. We present a library of computer codes, benchmark instances, and pointers to relevant articles for these two problems. The library is available at http://or.dei.unibo.it/library/bpplib. The computer code section includes twelve programs: seven are directly downloadable from the library page, while for the remaining five we provide addresses where they can be obtained or downloaded. Some of the codes for which we provide an original C++ implementation need an integer linear programming solver. For such cases, the library provides two versions: one that uses the commercial solver CPLEX, and one that uses the freeware solver SCIP. The benchmark section provides over six thousands instances (partly coming from the literature and partly randomly generated), together with the corresponding solutions. Instances that are difficult to solve to proven optimality are included. The library also includes a BibTeX file of more than 150 references on this topic and an interactive visual tool to manually solve bin packing and cutting stock instances. We conclude this work by reporting the results of new computational experiments on a number of computer codes and benchmark instances.

Keywords Bin packing · Cutting stock · Computer codes · Benchmark instances · Surveys

✉ Silvano Martello
silvano.martello@unibo.it

Maxence Delorme
maxence.delorme2@unibo.it

Manuel Iori
manuel.iori@unimore.it

1 DEI "Guglielmo Marconi", University of Bologna, Bologna, Italy

2 DISMI, University of Modena and Reggio Emilia, Modena, Italy

1 Introduction

In the *bin packing problem* (BPP), n *items* of given integer *weight* w_j $(j = 1, \ldots, n)$ have to be packed into the minimum number of identical containers (*bins*) of integer *capacity* c. Let u be any upper bound on the solution value. Let us introduce two sets of binary variables: y_i $(i = 1, \ldots, u)$, taking the value one if and only if bin i is used in the solution, and x_{ij} $(i = 1, \ldots, u;\ j = 1, \ldots, n)$, taking the value one if and only if item j is packed into bin i. A possible simple *Integer linear programming* (ILP) model of the problem is then (see Martello and Toth [28])

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{u} y_i && && (1)\\
\text{s.t.} \quad & \sum_{j=1}^{n} w_j x_{ij} \le c y_i && (i = 1, \ldots, u), && (2)\\
& \sum_{i=1}^{u} x_{ij} = 1 && (j = 1, \ldots, n), && (3)\\
& y_i \in \{0, 1\} && (i = 1, \ldots, u), && (4)\\
& x_{ij} \in \{0, 1\} && (i = 1, \ldots, u;\ j = 1, \ldots, n). && (5)
\end{aligned}
$$

Among the many variants and generalizations of the problem, the most intensively studied is probably the *Cutting stock problem* (CSP). In this case, instead of single items, we have m *item types* of weight w_j and an integer *demand* d_j $(j = 1, \ldots, m)$ per item type. The objective is to pack d_j copies of each item type j into the minimum number of bins. By replacing binary variables x_{ij} with a set of integer variables ξ_{ij} $(i = 1, \ldots, u;\ j = 1, \ldots, m)$ giving the number of items of type j packed into bin i, the CSP can be modeled by the ILP

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{u} y_i && && (6)\\
\text{s.t.} \quad & \sum_{j=1}^{m} w_j \xi_{ij} \le c y_i && (i = 1, \ldots, u), && (7)\\
& \sum_{i=1}^{u} \xi_{ij} = d_j && (j = 1, \ldots, m), && (8)\\
& y_i \in \{0, 1\} && (i = 1, \ldots, u), && (9)\\
& \xi_{ij} \ge 0, \text{ integer} && (i = 1, \ldots, u;\ j = 1, \ldots, m). && (10)
\end{aligned}
$$

The BPP is known to be $\mathcal{NP}$-hard in the strong sense (by transformation from the 3-Partition problem, see Garey and Johnson [20]). As any instance of either problem can easily be transformed into an equivalent instance of the other, the same holds for the CSP.

These two problems are among the most intensively studied problems in combinatorial optimization. Two recent surveys on exact methods (Delorme et al. [14]) and approximation algorithms (Coffman et al. [10]) consider in total over 230 different references. Previous surveys were presented by Garey and Johnson [21], Coffman et al. [11,12], Sweeney and Paternoster [35], Dyckhoff [16], Martello and Toth [28] (Chapter 8), Dyckhoff and Finke [17], Valério de Carvalho [38], Wäscher et al. [40], among others. Most solution methodologies have been tried on these problems: different kinds of ILP models, lower bound computations, branch-and-bound, branch-and-price, constraint programming, approximation algorithms, heuristics, and metaheuristics.

In the next section we list a number of relevant web-based libraries for optimization problems. In Sect. 3 we introduce the computer codes and the visual solver provided by the BPPLIB. In Sect. 4 we describe the available benchmarks: some of them were used in [14] for the computational evaluations of the different exact approaches, using commercial solver CPLEX when needed. As the library has been enriched by also providing versions based on the freeware solver SCIP, in Sect. 5 we provide new experiments aiming at evaluating the computational difference between the two versions. In addition, we describe new test instances, that appeared after the publication of [14], and present the corresponding computational experiments.

2 Web-based libraries for optimization problems

A number of web-based libraries dedicated to different optimization problems can be found on the Internet. The oldest one is probably the famous

- **OR-Library**, a collection of test data sets for a variety of operations research problems (including one- and two-dimensional packing problems), implemented by Beasley [2], see http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

More specific relevant libraries are, among others:

- **QAPLIB**, implemented by Burkard et al. [6,7] for the *Quadratic assignment problem*, see http://anjos.mgi.polymtl.ca/qaplib/. It provides instances, relevant references, and computer codes;
- **Traveling salesman problem web page**, created by Applegate et al. [1], see http://www.math.uwaterloo.ca/tsp/. It contains instances, computer codes, pointers to the literature, and educational tools;
- **VRPH**, a library of heuristics for the *Vehicle routing problems* (VRP), authored by Groër et al. [23], see http://sites.google.com/site/vrphlibrary/;
- **MIPLIB**, a *Mixed integer programming* library, providing a large set of instances, created by Koch et al. [25], see http://miplib.zib.de/;
- **libcgrpp**, a library for *Bound-constrained global optimization*, implemented by Silva et al. [34], see http://www.swmath.org/software/7205, containing a number of computer codes;
- **CBLIB 2014**, a collection of benchmark problems for *Conic mixed-integer and continuous optimization*, constructed by Friberg [19], see http://cblib.zib.de/;
- **CVRPLIB**, a repository of VRP benchmark instances and solutions, designed by Uchoa et al. [36], see http://vrp.atd-lab.inf.puc-rio.br/. It also proposes rewards for the solution of challenging instances.

We present here the BPPLIB, a library dedicated to *Bin Packing* and *Cutting stock problems*, available at http://or.dei.unibo.it/library/bpplib. An earlier, smaller version of the library was implemented as an auxiliary instrument for the computational experiments presented in [14]. The current BPPLIB contains pointers to the literature, original computer codes, links to computer codes from the Internet, benchmark instances, and an open source visual application to interactively solve BPP instances. Data sets and problem generators for the BPP and the CSP can also be found at the page of ESICUP (the EURO working group on cutting and packing), https://paginas.fe.up.pt/~esicup/about.

3 Computer codes

The BPPLIB provides twelve computer codes of different types for the exact solution of the BPP and the CSP. The choice of such codes was motivated by a number of properties: historical relevance, efficiency, reliability, and availability of the corresponding computer codes. The main characteristics of the codes are summarized in Table 1. More detailed information is provided below.

3.1 Branch-and-bound

Implicit enumeration has been the first tool for the study of methods for optimally solving the BPP. The first effective exact algorithms for the BPP were indeed based on a branch-and-bound approach. The library provides, in chronological order:

- **MTP**: Fortran code of the BPP algorithm by Martello and Toth [28], originally available in the diskette accompanying the book. The algorithm adopts a depth-first strategy to explore a branch–decision tree that considers one item per level: descendant nodes are generated by assigning the current item, in turn, to all already initialized bins and possibly to a new bin. While the approach is effective for BPP instances, considering one item at a time is clearly inefficient for CSP instances with high item multiplicity. The code can be run using the Fortran front end of the GNU compiler collection GCC;
- **BISON**: Scholl et al. [32] obtained a very efficient BPP algorithm by enriching MTP through new lower bounds and a Tabu search algorithm to help the search by means of effective heuristic solutions. The code was implemented in Pascal, and can be obtained from the authors, using the address provided in the library. Worth is mentioning that, in spite of its 'age', this program is still working and quite effective: at the time of writing, it can be run using compiler `fpc (version 3.0.0 for x86_64)`;
- **CVRPSEP**: we provide a link to the C code implemented by J. Lysgaard as part of a separation routine within the algorithm by Lysgaard et al. [27] for the capacitated vehicle routing problem. The routine was obtained by using procedures from MTP. It is generally less efficient than MTP, but we decided to include it in the library mainly because one may prefer a C code to a Fortran code. The implementation details can be found in a technical report by Lysgaard [26].

Table 1 Main characteristics of the computer codes provided by the BPPLIB

Code	Problem	BPPLIB	Language	Type	References	Computer code		Supported ILP solver		
						Author	Year	CPLEX	Gurobi	Scip
MTP	BPP	Code	Fortran	B&B	[28]	Martello and Toth	1990	Not required		
BISON	BPP	Pointer	Pascal	B&B	[32]	Scholl et al.	1997	Not required		
CVRPSEP	BPP	Link	C	B&B	[27]	Lysgaard	2004	Not required		
BELOV	CSP	Link	C++	B&C&P	[3]	Belov	2006	x		
SCIP-BP	BPP	Link	C	B&P	[30]	Berthold and Heinz	c. 2010			x
ONECUT	CSP	Code	C++	ILP	[15]	Delorme	2014	x		x
ARCFLOW	CSP	Code	C++	ILP	[37]	Delorme	2014	x		x
DPFLOW	BPP	Code	C++	ILP	[8]	Delorme	2014	x		x
VPSOLVER	CSP	Link	C++	ILP	[5]	Brandão	2014	x	x	x

3.2 Branch-and-price

Branch-and-price is the modern evolution of branch-and-bound, obtained by combining implicit enumeration and column generation. The resulting approach can produce very effective algorithms for the problems at hand. We provide links to two computer codes:

- **BELOV**: C++ implementation by G. Belov of the branch-and-cut-and-price algorithm by Belov and Scheithauer [3], using CPLEX for the inner routines. The algorithm is tailored to the exact solution of CSP instances, and it computationally proved to be the most powerful approach both in the case of low and high item multiplicity;
- **SCIP-BP**: freeware SCIP C code for a branch-and-price BPP algorithm based on the classical Ryan and Foster [30] branching rule and available at the SCIP web page. This code is only effective for instances with small number of item types and low item multiplicity.

3.3 Pseudo-polynomial formulations solved via ILP

Already in the Seventies, pseudo-polynomial models coming from a graph representation of the solution space were proposed. For many years, solution approaches based on such models have been regarded as very theoretical, with no practical interest, due to the huge number of variables and constraints they imply. Up to few years ago, these methods were mainly used within branch-and-price algorithms (see, e.g., Valério de Carvalho [37], who proposed to solve his pseudo-polynomial model through column generation). However, nowadays computational power of ILP solvers made them competitive with branch-and-price algorithm also for the case of realistic size instances, provided the number of generated variables (that depends on capacity, number of items, and item weights) is not too big. The BPPLIB provides four algorithms based on pseudo-polynomial models:

- **ONECUT**: C++ implementation of the *one-cut* CSP model independently defined in the Seventies by Rao [29], and Dyckhoff [15];
- **ARCFLOW**: C++ implementation of the *arc-flow* CSP model by Valério de Carvalho [37];
- **DPFLOW**: C++ implementation of the *DP-flow* BPP model by Cambazard and O'Sullivan [8];
- **VPSOLVER**: link to the C++ implementation by Brandão and Pedroso [5] of their CSP algorithm. This is currently the most effective pseudo-polynomial approach, and its performance is often competitive with that of BELOV.

For the first three codes we provide both a version that uses CPLEX as an inner routine, and a version that uses SCIP, together with the corresponding makefiles. Code VPSOLVER was instead implemented by the authors in a version that invokes Gurobi.

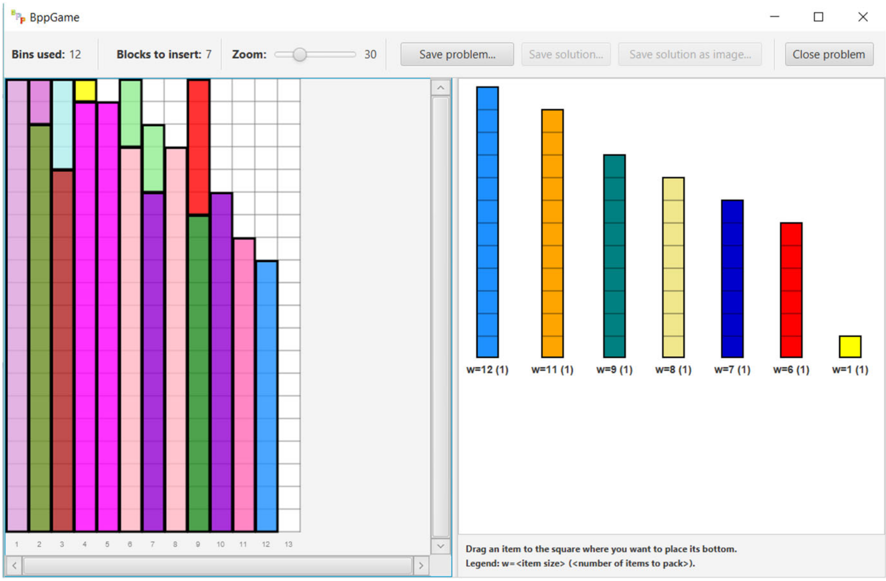

Fig. 1 The interactive visual solver

3.4 BppGame: an interactive visual solver

The library includes the pointer to an open source visual ScalaFX application to interactively solve BPP and CSP instances. The application is derived from a more general tool for the solution of two-dimensional packing problems, see Costa et al. [13]. It allows an easy interaction to obtain a feasible solution of a given problem instance. The application has a number of features, that are fully described in its own web page http://gianlucacosta.info/BppGame/. The easiest way to test it consists in following the hyperlink and executing the sequence of actions: Download zip and extract its contents → BppGame-x.x → bin → BppGame.bat (Windows) or BppGame (Linux) → Sample problems. Figure 1 shows a BppGame visualization. The user can click on an item on the right frame, and drag and drop it to a selected position in the left frame.

4 Benchmarks

The BPPLIB provides 6195 benchmark instances belonging to four categories. Each instance is provided, using unified formats, both in BPP and CSP version. The main characteristics of the instances are summarized in Table 2. Detailed information is provided below.

Table 2 Main characteristics of the instances provided by the BPPLIB

Instances	References	Parameters of the instances				Specificity		
		#	n	c	Distribution	Large c	Perf. pack.	Non-IRUP
Falkenauer U	[18]	80	{120, 250, 500, 1000}	150	Uniform			
Falkenauer T	[18]	80	{60, 120, 249, 501}	1000	Ad-hoc		x	
Scholl1	[32]	720	{50, 100, 200, 500}	{100, 120, 150}	Uniform			
Scholl2	[32]	480	{50, 100, 200, 500}	1000	Uniform			
Scholl3	[32]	10	200	100,000	Uniform	x		
Wascher	[39]	17	[57–239]	10,000	Ad-hoc	x	x	x
Schwerin1	[33]	100	100	1000	Uniform			
Schwerin2	[33]	100	120	1000	Uniform			
Hard28	[31]	28	{160, 180, 200}	1000	Ad-hoc		x	x
Random	[14]	3840	{50,100,200,...,500,750,1000}	{50, 75, 100, 120,...,750, 1000}	Uniform			
AI	[14]	250	{202, 403, 601, 802, 1003}	$\leq${2500, 10K, 20K, 40K, 80K}	Ad-hoc	x	x	
ANI	[14]	250	{201, 402, 600, 801, 1002}	$\leq${2500, 10K, 20K, 40K, 80K}	Ad-hoc	x		x
GI	[24]	240	$\sim$10.5×{125,250,500}	{500,000, 1,500,000}	Uniform	x		

4.1 Literature instances

This section contains the 1615 instances proposed by

- Falkenauer [18]: 80 (easy) instances with uniformly distributed item sizes and 80 (more difficult) instances obtained through triplets of items that, in any optimal solution, must be packed into the same bin wihtout leaving unused space (*perfect packing*).
- Scholl et al. [32]: three sets of instances with uniformly distributed item sizes. The first set is composed by 720 easy instances, the second set by 480 instances of medium difficulty, and the third set by 10 difficult instances characterized by huge capacities;
- Wäscher and Gau [39]: 17 very hard instances selected by the authors from a much larger set of instances belonging to different typologies;
- Schwerin and Wäscher [33]: two sets of 100 relatively easy instances each;
- Schoenfield [31]: 28 hard instances that do not involve huge capacities.

In the library, each set is identified by the name of the (first) author. Additional instances can be created through the provided link to the instance generators proposed by Schwerin and Wäscher [33] and Gau and Wäscher [22].

4.2 Randomly generated instances

The library provides the 3840 instances that were randomly generated for the computational experiments reported in [14]. The instances have different values of n (50, 100, 200, 300, 400, 500, 750, 1000), of c (50, 75, 100, 120, 125, 150, 200, 300, 400, 500, 750, 1000), and of the minimum ($0.1\,c$, $0.2\,c$) and maximum ($0.7\,c$, $0.8\,c$) item weight. The benchmark contains 10 instances for each of the 384 quadruplets (n, c, minimum weight, maximum weight). These instances are relatively easy, and the algorithms listed in Sect. 3 could solve most of them within reasonable CPU times.

4.3 Hard instances

An instance of an optimization problem that possesses the so-called *Integer round-up property* (see Berge and Johnson [4]) is called an IRUP instance and is generally considered less difficult to solve in practice with respect to instances not possessing such property. In order to perform experiments on challenging instances, a number of so called *augmented Non-IRUP* and *augmented IRUP* instances were proposed in [14], using as a basis a set of Non-IRUP instances presented in Caprara et al. [9]. For the 250 instances of the former class an optimal solution is easy to find, but its optimality is very difficult to prove. Even the continuous relaxation of the set covering formulation (the basis of branch-and-price algorithms) and that of the pseudo-polynomial formulations fail in reaching the optimal value. As a consequence, algorithms based on such relaxations require either a huge branching process or a heavy cut generation: already for $n \approx 400$, no algorithm is capable of solving all of them to proven optimality. For the 250 instances of the latter class, it is easy to produce a lower bound whose value is equal to the optimum, but it is difficult to build an optimal solution.

4.4 GI instances

The library includes 240 new instances, recently proposed by Gschwind and Irnich [24]. Such instances, uniformly randomly generated, are characterized by very large capacities. They are organized into four sets of 60 instances each. As shown in the next section, two of such sets are generally difficult to solve.

5 Computational experiments

We report the results of some experiments executed on an Intel Xeon 3.10 GHz (equipped with four cores) with 8 GB RAM, all executed with a single core. In order to test the codes on *non-trivial* instances, we preliminarily obtained an upper bound through the classical *best fit decreasing heuristic* and computed the lower bound value known as L2 (see [24]): only instances for which these two values were different were then tested.

Tables 3 and 4 give the number of literature instances that were solved in 1 CPU min (and, in parentheses, the average CPU time), by, respectively, the enumeration algorithms and the pseudo-polynomial models (for the non-solved instances, 1 CPU min was considered). For each instance set, boldface highlights the cases where all instances were solved to proven optimality.

The results in Table 3 summarize the (much more detailed) tables presented in [14]: they are provided here in order to give the reader information on the performance of the codes provided in the BPPLIB. The results in Table 4 include new results obtained

Table 3 Benchmarks from the literature (non-trivial instances only), enumerative algorithms

Set	Number of tested instances	Branch-and-bound			Branch-and-price	
		MTP	BISON	CVRPSEP	BELOV	SCIP-BP
Falkenauer U	74	22 (42.8)	44 (24.5)	22 (42.2)	**74** (0.0)	18 (50.1)
Falkenauer T	80	6 (55.5)	42 (30.6)	0 (60.0)	57 (24.7)	35 (39.4)
Scholl1	323	242 (15.1)	288 (7.0)	223 (19.4)	**323** (0.0)	244 (22.4)
Scholl2	244	130 (28.2)	233 (3.0)	65 (44.2)	**244** (0.3)	67 (49.2)
Scholl3	10	0 (60.0)	3 (42.0)	0 (60.0)	**10** (14.1)	0 (60.0)
Wäscher	17	0 (60.0)	10 (24.7)	0 (60.0)	**17** (0.1)	0 (60.0)
Schwerin1	100	15 (51.1)	100 (0.0)	9 (55.4)	**100** (1.0)	0 (60.0)
Schwerin2	100	4 (57.6)	63 (22.2)	0 (60.0)	**100** (1.4)	0 (60.0)
Hard28	28	0 (60.0)	0 (60.0)	0 (60.0)	**28** (7.3)	7 (51.2)
Total (average)	976	419 (34.4)	783 (12.3)	319 (40.8)	953 (2.7)	371 (42.2)

Number of instances solved in less than 1 min (average CPU time in s)

Table 4 Benchmarks from the literature (non-trivial instances only), pseudo-polynomial models

Set	Number of tested instances	ONECUT		ARCFLOW		DPFLOW		VPSOLVER
		CPLEX	SCIP	CPLEX	SCIP	CPLEX	SCIP	
Falkenauer U	74	**74** (0.2)	67 (23.8)	**74** (0.2)	70 (18.7)	37 (38.8)	0 (60.0)	**74** (0.1)
Falkenauer T	80	**80** (8.7)	21 (44.9)	**80** (3.5)	33 (41.4)	40 (41.7)	20 (50.8)	**80** (0.4)
Scholl1	323	**323** (0.1)	318 (5.0)	**323** (0.1)	320 (5.1)	289 (13.0)	178 (34.0)	**323** (0.1)
Scholl2	244	118 (38.7)	20 (56.3)	202 (18.9)	39 (53.7)	58 (50.4)	11 (58.5)	208 (14.0)
Scholl3	10	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	**10** (6.3)
Wäscher	17	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	6 (49.4)
Schwerin1	100	**100** (13.1)	0 (60.0)	**100** (1.5)	0 (60.0)	0 (60.0)	0 (60.0)	**100** (0.3)
Schwerin2	100	**100** (11.7)	0 (60.0)	**100** (1.5)	1 (59.5)	0 (60.0)	0 (60.0)	**100** (0.3)
Hard28	28	6 (54.6)	0 (60.0)	16 (40.6)	0 (60.0)	0 (60.0)	0 (60.0)	27 (14.2)
Total (average)	976	801 (16.3)	426 (36.9)	895 (8.2)	463 (35.6)	424 (38.9)	209 (50.3)	928 (5.0)

Number of instances solved in less than 1 min (average CPU time in s)

Table 5 Randomly generated benchmarks (non-trivial instances only), enumerative algorithms

n	Number of tested instances	Branch-and-bound			Branch-and-price	
		MTP	BISON	CVRPSEP	BELOV	SCIP-BP
50	165	163 (0.8)	**165** (0.0)	164 (0.4)	**165** (0.0)	**165** (0.9)
100	271	243 (7.4)	257 (3.8)	239 (8.4)	**271** (0.0)	**271** (4.6)
200	359	237 (21.6)	290 (12.0)	220 (25.0)	**359** (0.0)	293 (22.6)
300	393	166 (35.7)	265 (20.7)	144 (38.7)	**393** (0.1)	155 (44.1)
400	425	151 (39.1)	244 (26.1)	138 (41.2)	**425** (0.2)	114 (49.8)
500	414	121 (43.0)	208 (30.3)	128 (42.6)	**414** (0.2)	69 (55.1)
750	433	93 (47.3)	214 (30.9)	98 (47.3)	**433** (0.4)	22 (59.5)
1000	441	78 (49.5)	196 (33.9)	73 (50.8)	**441** (0.7)	0 (60.0)
Total (average)	2901	1252 (34.7)	1839 (22.6)	1204 (36.0)	**2901** (0.2)	1089 (42.4)

Number of instances solved in less than 1 min (average CPU time in s)

using SCIP as the ILP solver. The tables confirm the clear superiority of BELOV and VPSOLVER over the other algorithms.

Table 4 shows in addition that the performance of ONECUT and ARCFLOW is not affected by the ILP solver for most of the easy instances, while their performance for the difficult instances sharply worsens when SCIP is used instead of CPLEX. This behavior could be explained by the difference in the number of generated variables and constraints between different benchmarks. For example, ARCFLOW produces, on average, 1735 variables and 103 constraints for "Scholl 1" instances, while for "Scholl 2" it produces, on average, 39,307 variables and 840 constraints.

Tables 5 and 6 refer to the randomly generated instances and provide the same information as in Tables 3 and 4. The previous observations are confirmed: BELOV and VPSOLVER outperform the other approaches, and the use of SCIP decreases the algorithms' performance, especially for large values of n.

We report in Table 7 the results of computational experiments for the GI benchmark, a set of CSP instances recently proposed by Gschwind and Irnich [24] for testing their dual inequalities aimed at stabilizing column generation processes. They are organized into four groups (AA, AB, BA, and BB), characterized by different item weight ranges and capacities. Each group has three sets of 20 instances each, characterized by the number of item types (125, 250, and 500).

We tested the best enumerative algorithm (BELOV) and the best pseudo-polynomial approaches (ARCFLOW and VPSOLVER) with a time limit of 1 h. BELOV could solve all of these instances very quickly, while they turned out to be extremely difficult for the pseudo-polynomial models. The behavior of the latter approaches was particularly poor for the instances that have items with very small weight and huge capacities (AB and BB, with $c \geq 500{,}000$), which induce a high number of variables and constraints. For example, the ILP model produced by ARCFLOW has on average 549,441 variables and 131,219 constraints for instances AA with $m = 125$, but 5,754,617 variables and 404,283 constraints for instances AB with $m = 125$.

Table 6 Randomly generated benchmarks (non-trivial instances only), pseudo-polynomial models

n	Number of tested instances	ONECUT		ARCFLOW		DPFLOW		VPSOLVER
		CPLEX	SCIP	CPLEX	SCIP	CPLEX	SCIP	
50	165	**165** (0.1)	163 (2.0)	**165** (0.1)	**165** (1.6)	**165** (0.5)	162 (5.1)	**165** (0.0)
100	271	**271** (0.8)	249 (8.6)	**271** (0.3)	262 (10.1)	**271** (5.0)	168 (34.5)	**271** (0.1)
200	359	358 (2.4)	286 (15.4)	**359** (0.8)	278 (20.2)	292 (21.0)	76 (51.8)	**359** (0.3)
300	393	385 (4.5)	272 (22.2)	391 (2.0)	262 (24.9)	243 (33.9)	31 (57.3)	**393** (0.6)
400	425	408 (5.1)	293 (22.0)	421 (3.0)	276 (25.8)	193 (42.4)	23 (58.1)	**425** (0.8)
500	414	394 (6.3)	275 (24.0)	402 (4.0)	258 (26.5)	169 (44.8)	13 (58.8)	413 (1.7)
750	433	401 (7.8)	284 (24.3)	415 (6.0)	279 (25.7)	120 (52.6)	12 (59.1)	431 (2.4)
1000	441	407 (8.1)	280 (25.8)	416 (6.8)	281 (26.1)	67 (56.4)	7 (59.6)	434 (3.4)
Total (average)	2901	2789 (5.0)	2102 (20.0)	2840 (3.3)	2061 (22.3)	1520 (36.7)	492 (52.5)	2891 (1.4)

Number of instances solved in less than 1 min (average CPU time in s)

Table 7 Number of GI instances solved in less than 1 h (average time in s)

Set	m	Number of tested instances	BELOV	ARCFLOW	VPSOLVER
AA	125	20	**20** (0.1)	19 (1092.6)	**20** (0.9)
	250	20	**20** (0.9)	0 (3600.0)	**20** (14.5)
	500	20	**20** (7.5)	0 (3600.0)	16 (1345.9)
AB	125	20	**20** (0.7)	0 (3600.0)	0 (3600.0)
	250	20	**20** (2.1)	0 (3600.0)	0 (3600.0)
	500	20	**20** (29.9)	0 (3600.0)	0 (3600.0)
BA	125	20	**20** (0.1)	**20** (1120.9)	**20** (1.4)
	250	20	**20** (1.3)	0 (3600.0)	**20** (23.1)
	500	20	**20** (7.2)	0 (3600.0)	17 (1450.2)
BB	125	20	**20** (0.2)	0 (3600.0)	0 (3600.0)
	250	20	**20** (2.3)	0 (3600.0)	0 (3600.0)
	500	20	**20** (29.1)	0 (3600.0)	0 (3600.0)
Total (average)		240	**240** (6.8)	39 (3234.8)	113 (2036.3)

6 Conclusions and future research

We presented the BPPLIB, a library dedicated to Bin Packing and Cutting Stock Problems that provides computer codes, benchmark instances, pointers to relevant research papers, a bibliography, and an interactive visual solver. The computer code section includes twelve exact programs that make use of different optimization paradigms. To fit the needs of both researchers and practitioners, we provide two versions of the programs that need an ILP solver: one using the commercial solver CPLEX and the other using the freeware solver SCIP. The benchmark section provides over six thousands instances, among which a few hundreds remain unsolved to proven optimality despite the application of all available exact methods.

We believe that the BPPLIB can be a useful tool to foster new research on the challenging area of Bin Packing and Cutting Stock optimization. In addition, it can be used for educational purposes as it provides an interactive tool (the "BppGame", see Fig. 1) that students can use to easily understand the features and the difficulty of optimization problems.

As future activity, we plan to maintain the library updated with new relevant contributions that will appear on these problems, as well as to expand it through the most interesting problem variants.

Acknowledgements Research supported by Air Force Office of Scientific Research (Grant FA9550-17-1-0067) and by MIUR-Italy (Grant PRIN 2015). We thank Gianluca Costa for the development of the BppGame. We thank the reviewers for useful comments.

References

1. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem—A Computational Study. Princeton University Press, Princeton (2006)
2. Beasley, J.E.: OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**, 1069–1072 (1990)
3. Belov, G., Scheithauer, G.: A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. Eur. J. Oper. Res. **171**, 85–106 (2006)
4. Berge, C., Johnson, E.L.: Coloring the edges of a hypergraph and linear programming techniques. Ann. Discrete Math. **1**, 65–78 (1977)
5. Brandão, F., Pedroso, J.P.: Bin packing and related problems: general arc-flow formulation with graph compression. Comput. Oper. Res. **69**, 56–67 (2016)
6. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB—a quadratic assignment problem library. Eur. J. Oper. Res. **55**, 115–119 (1991)
7. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB—a quadratic assignment problem library. J. Global Optim. **10**, 391–403 (1997)
8. Cambazard, H., O'Sullivan, B.: Propagating the bin packing constraint using linear programming. In: Principles and Practice of Constraint Programming—CP 2010, Volume 6308 of Lecture Notes in Computer Science, pp. 129–136. Springer, Berlin (2010)
9. Caprara, A., Dell'Amico, M., Díaz Díaz, J.C., Iori, M., Rizzi, R., Rizzi, R.: Friendly bin packing instances without integer round-up property. Math. Program. **150**, 5–17 (2014)
10. Coffman Jr., E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: survey and classification. In: Pardalos, P.M., Du, D.-Z., Graham, R.L. (eds.) Handbook of Combinatorial Optimization. Springer, New York (2013)
11. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin-packing—an updated survey. In: Ausiello, G., Lucentini, M., Serafini, P. (eds.) Algorithm Design for Computer System Design, pp. 49–106. Springer, Vienna (1984)
12. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 46–93. PWS Publishing Co., Boston (1996)
13. Costa, G., Delorme, M., Iori, M., Malaguti, E., Martello, S.: A training software for orthogonal packing problems. Comput. Ind. Eng. **111**, 139–147 (2017)
14. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: mathematical models and exact algorithms. Eur. J. Oper. Res. **255**, 1–20 (2016)
15. Dyckhoff, H.: A new linear programming approach to the cutting stock problem. Oper. Res. **29**, 1092–1104 (1981)
16. Dyckhoff, H.: A typology of cutting and packing problems. Eur. J. Oper. Res. **44**, 145–159 (1990)
17. Dyckhoff, H., Finke, U.: Cutting and Packing in Production and Distribution. Physica-Verlag, Heidelberg (1992)
18. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. J. Heuristics **2**, 5–30 (1996)
19. Friberg, H.A.: CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization. Math. Program. Comput. **8**, 191–214 (2016)
20. Garey, M.G., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
21. Garey, M.R., Johnson, D.S.: Approximation algorithms for bin-packing problems: a survey. In: Ausiello, G., Lucertini, M. (eds.) Analysis and Design of Algorithms in Combinatorial Optimization, pp. 147–172. Springer, Vienna (1981)
22. Gau, T., Wäscher, G.: CUTGEN1: a problem generator for the standard one-dimensional cutting stock problem. Eur. J. Oper. Res. **84**(3), 572–579 (1995)
23. Groër, C., Golden, B., Wasil, E.: A library of local search heuristics for the vehicle routing problem. Math. Program. Comput. **2**, 79–101 (2010)
24. Gschwind, T., Irnich, S.: Dual inequalities for stabilized column generation revisited. INFORMS J. Comput. **28**, 175–194 (2016)
25. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**, 103 (2011)

26. Lysgaard, J.: CVRPSEP: a package of separation routines for the capacitated vehicle routing problem. Technical Report, Aarhus School of Business (2003)
27. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math. Program. **100**, 423–445 (2004)
28. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester (1990). http://www.or.deis.unibo.it/knapsack.html
29. Rao, M.R.: On the cutting stock problem. J. Comput. Soc. India **7**, 35–39 (1976)
30. Ryan, D.M., Foster, B.A.: An integer programming approach to scheduling. In: Wren, A. (ed.) Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling, pp. 269–280. North-Holland, Amsterdam (1981)
31. Schoenfield, J.E.: Fast, exact solution of open bin packing problems without linear programming. Technical Report, US Army Space and Missile Defense Command, Huntsville (2002)
32. Scholl, A., Klein, R., Jürgens, C.: Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput. Oper. Res. **24**, 627–645 (1997)
33. Schwerin, P., Wäscher, G.: The bin-packing problem: a problem generator and some numerical experiments with FFD packing and MTP. Int. Trans. Oper. Res. **4**, 377–389 (1997)
34. Silva, R.M.A., Resende, M.G.C., Pardalos, P.M., Hirsch, M.J.: A python/C library for bound-constrained global optimization with continuous GRASP. Optim. Lett. **7**(5), 967–984 (2013)
35. Sweeney, P.E., Paternoster, E.R.: Cutting and packing problems: a categorized, application-orientated research bibliography. J. Oper. Res. Soc. **43**, 691–706 (1992)
36. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. **257**, 845–858 (2017)
37. Valério de Carvalho, J.M.: Exact solution of bin-packing problems using column generation and branch-and-bound. Ann. Oper. Res. **86**, 629–659 (1999)
38. Valério de Carvalho, J.M.: LP models for bin packing and cutting stock problems. Eur. J. Oper. Res. **141**, 253–273 (2002)
39. Wäscher, G., Gau, T.: Heuristics for the integer one-dimensional cutting stock problem: a computational study. Oper. Res. Spektrum **18**, 131–144 (1996)
40. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. **183**, 1109–1130 (2007)