



HAL
open science

New Symmetry-less ILP Formulation for the Classical One Dimensional Bin-Packing Problem

Khadija Hadj Salem, Yann Kieffer

► **To cite this version:**

Khadija Hadj Salem, Yann Kieffer. New Symmetry-less ILP Formulation for the Classical One Dimensional Bin-Packing Problem. Donghyun Kim; R. N. Uma; Zhipeng Cai; Dong Hoon Lee. Computing and Combinatorics. 26th International Conference, COCOON 2020, Atlanta, GA, USA, August 29–31, 2020, Proceedings, 12273, Springer, pp.423-434, 2020, Lecture Notes in Computer Science, 978-3-030-58149-7. 10.1007/978-3-030-58150-3_34 . hal-03006106

HAL Id: hal-03006106

<https://hal.science/hal-03006106>


Submitted on 18 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



New Symmetry-less ILP Formulation for the Classical One Dimensional Bin-Packing Problem

Khadija Hadj Salem¹(✉)  and Yann Kieffer²

¹ Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002,
64 Avenue Jean Portalis, 37200 Tours, France

khadija.hadj-salem@univ-tours.fr

² Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France
yann.kieffer@lcis.grenoble-inp.fr

Abstract. In this article, we address the classical *One-Dimensional Bin Packing Problem* (1D-BPP), an \mathcal{NP} -hard combinatorial optimization problem. We propose a new formulation of integer linear programming for the problem, which reduces the search space compared to those described in the literature, as well as two families of cutting planes. Computational experiments are conducted on the data-set found in BPPLib and the results show that it is possible to solve more instances and to decrease the computation time by using our new formulation.

Keywords: Bin packing · Integer linear programming · Cutting plane

1 Introduction

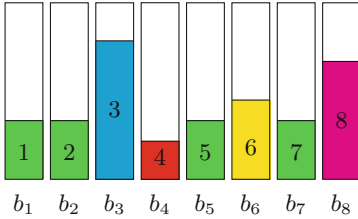
The *one-dimensional Bin Packing Problem*, noted 1D-BPP from here on, is a well studied combinatorial optimization problem, with a rich literature detailing different approaches for its solution. It can be informally defined as follows: n items have to be packed each into one of n available bins. Each item i has a non-negative weight w_i ($i = 1, \dots, n$) and all bins have the same positive integer capacity C . The objective is to find a packing with a minimum number of bins such that the total weights of the items in each bin does not exceed the capacity C .

We consider the following example, using a set of bins with capacity $C = 6$, a set of items $i = 1, 2, \dots, 8$, with weights w_i (given in Table 1). A feasible solution as well as an optimal solution, respectively, with 8 bins and 4 bins, are given in Fig. 1.

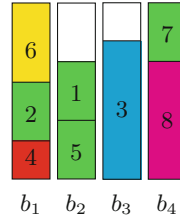
A central theme for this study is the computational effect of the removal of symmetric solutions. To the best of our knowledge, no numerical studies have been published to ascertain the performance gain of symmetry breaking constraints for 1D-BPP.

Table 1. An example of data, with 8 items

Items i	1	2	3	4	5	6	7	8
Weights w_i	2	2	5	1	2	3	2	4



A feasible solution, with 8 bins



An optimal solution, with 4 bins

Fig. 1. Solutions for the 1D-BPP

This article presents our study of a new symmetry-less formulation for 1D-BPP without and with adding cutting planes, and a comparative study of the performances of these two kinds of approaches.

The remainder of this paper is structured as follows. In the next section, we briefly review the exact solution methods for 1D-BPP which rely on integer linear programming. In Sect. 3, we present the new symmetry-less ILP formulation. Sect. 4 is devoted to cutting planes for that formulation. Finally, in Sect. 5, we present computational results obtained by running the proposed formulations on a number of benchmark instances for the 1D-BPP and discuss their performances. Conclusion and perspectives follow in Sect. 6.

2 Previous Work on ILP Formulations for 1D-BPP

2.1 Assignment-Based Models

The compact ILP formulation for 1D-BPP, which Martello and Toth attribute to Kantorovich (see [8]), is the following. Let y_j be a decision variable equal to 1 if bin j is used in the packing, and 0 otherwise, for all $j \in \{1, \dots, n\}$. Similarly, let x_{ij} be a decision variable equal to 1 if item i is packed into bin j , and 0 otherwise, for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\}$.

The full model, hereafter denoted as ILP-0, is:

$$\text{ILP} - 0 : \quad \min \sum_{j=1}^n y_j \tag{1}$$

$$\begin{cases}
 \sum_{j=1}^n x_{ij} = 1 & \forall i \in \{1, \dots, n\} & (2) \\
 \sum_{i=1}^n w_i * x_{ij} \leq C * y_j & \forall j \in \{1, \dots, n\} & (3) \\
 x_{ij} \in \{0, 1\} & \forall i \in \{1, \dots, n\}, j \in \{1, \dots, n\} & (4) \\
 y_j \in \{0, 1\} & \forall j \in \{1, \dots, n\} & (5)
 \end{cases}$$

In this formulation, constraints (2) ensure that each item is packed into exactly one bin, constraints (3) impose that the capacity of any used bin is not exceeded and both constraints (4) and (5) define the variable domains.

An obvious lower bound for the 1D-BPP, computable in $\mathcal{O}(n)$ time, is the optimal value of the *continuous relaxation* of ILP-0. This lower bound, usually denoted L_1 in the literature, can be computed by:

$$L_1 = \left\lceil \sum_{j=1}^n w_j / C \right\rceil \tag{6}$$

It is easily seen that the *worst-case performance ratio* of L_1 is equal to $\frac{1}{2}$ (see, e.g., [9]).

2.2 Other Methods for Optimally Solving the 1D-BPP

Among other methods for solving 1D-BPP exactly, we can find the pseudo-polynomial ILP formulations coming from a graph representation of the solution space and the branching algorithms. An overview of these methods is given in Table 2. More detailed information is provided below.

Table 2. An overview of exact solutions for the 1D-BPP: pseudo-polynomial models & branching algorithms

Methods	Type	Reference	Supported ILP solver
MTP	B&B	Martello and Toth (1990)	Not required
BISON	B&B	Scholl et al. (1997)	Not required
CVRPSEP	B&B	Lysgaard et al. (2004)	Not required
SCIP-BP	B&P	Ryan and Foster (1981)	SCIP ^a
ONECUT	ILP	Dyckhoff (1981)	CPLEX ^b
DPFLOW	ILP	Cambazard and O’Sullivan (2010)	CPLEX, SCIP
SchedILP	ILP	Arbib et al. (2017)	CPLEX

^aSCIP: Solving Constraint Integer Programs

^bCPLEX: <https://www.ibm.com/analytics/cplex-optimizer>

As shown in Table 2, the first four rows describe a set of four enumeration algorithms. Three of them are branch-and-bound algorithms proposed

by [8,11] and [7], respectively. The last one is a branch-and-price algorithm in [10]. In the same way, the last two rows give the two algorithms based on a pseudo-polynomial formulations solved through an ILP solver (like CPLEX, SCIP, GUROBI). The first one uses the model proposed by [5]. The second one uses the model proposed by [2]. The third one uses the model proposed by [1]. According to the results discussed in [3], among the approaches based on pseudo-polynomial models, the **DPFLOW** has mainly theoretical interest, but has the advantage of being easily understandable. In the same way, among the enumeration algorithms, the **SCIP-BP** is effective only on small-size instances ($n \leq 100$).

Computer codes of these methods can be found in the BPPLIB, a library dedicated to Bin Packing and Cutting Stock Problems, available at <http://or.dei.unibo.it/library/bpplib> [4].

3 A Symmetry-less ILP for the 1D-BPP

The study of the topic of symmetry breaking constraints for the classical formulation ILP-0 has led us to the consideration of an alternative encoding, and thus an alternate formulation for the 1D-BPP.

Instead of having variables encoding the membership of *items* in *bins*, this alternate encoding directly encodes a partition of the set of items. Indeed, a solution with k bins to an instance of the 1D-BPP can be seen as a partition of the n items into k parts. The actual parts of the partition are referenced by their smallest-indexed item. Only one set of doubly-indexed variables is necessary for this: in contrast to the ILP-0 formulation, no variable is used to represent the bins.

More precisely, the variable z_{ij} is set to 1 if the lowest-indexed item sharing the same bin as item i is item j , and 0 otherwise, for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, i\}$. Then $z_{ii} = 1$ if and only if i is the smallest-indexed item in its bin. One can think of bins as labeled by their smallest-indexed item. Hence, counting bins can be achieved by summing the diagonal variables z_{jj} . The set of constraints for this formulation includes all constraints of the classical formulation; only y_j has to be replaced by z_{jj} . We denote by ILP-1 the resulting formulation.

$$\text{ILP - 1 : } \min \sum_{j=1}^n z_{jj} \tag{7}$$

$$s.t. \left\{ \begin{array}{ll} \sum_{j=1}^i z_{ij} = 1 & \forall i \in \{1, \dots, n\} \tag{8} \\ \sum_{i=j}^n w_i * z_{ij} \leq C * z_{jj} & \forall j \in \{1, \dots, n\} \tag{9} \\ z_{ij} \in \{0, 1\} & \forall i \in \{1, \dots, n\}, j \in \{1, \dots, i\} \tag{10} \end{array} \right.$$

We now argue that this formulation admits no symmetry at all.

Theorem 1. *There is a one-to-one correspondence between the encodings of feasible solutions for ILP-1, and partitions of the items.*

Proof. Given an encoding $z = (z_{ij})_{i,j}$ of a feasible solution, we can recover the parts of the associated partition in this manner: there are as many parts as indices j for which $z_{jj} = 1$ and the set of these parts is $S_j = \{i \in \{j, \dots, n\} \mid z_{ij} = 1\}$.

Given a partition $\mathcal{P} = (S_j)_{j \in I}$ of $\{1, \dots, n\}$, the associated encoding is the one described in an earlier part of this section. \square

Using the same example given in Sect. 1, the optimal solution to ILP-0 given in Fig. 1 can be encoded as a solution to the formulation ILP-1, as shown in Fig. 2. In this solution, items 5 and 1 are packed in bin 1; items 2, 4 and 6 are packed in bin 2; item 3 is packed in bin 3 and finally items 7 and 8 are packed in bin 7.

	1	2	3	4	5	6	7	8
1	1	×	×	×	×	×	×	×
2	0	1	×	×	×	×	×	×
3	0	0	1	×	×	×	×	×
4	0	1	0	0	×	×	×	×
5	1	0	0	0	0	×	×	×
6	0	1	0	0	0	0	×	×
7	0	0	0	0	0	0	1	×
8	0	0	0	0	0	0	1	0

Fig. 2. A corresponding encoding for the optimal solution to ILP-0 using ILP-1

Since the ILP-1 admits no symmetry, one can expect that the solution of ILP-1 instances has better performance than equivalent ILP-0 instances of 1D-BPP. An empirical evaluation of that statement is actually part of the study that we report on in Sect. 5.

Another quality of formulation ILP-1 is that it is a strict ILP formulation, and it is compact. Hence, wherever an ILP formulation P includes a BPP-like set of constraints, akin to those found in ILP-0, these constraints can be replaced by those in ILP-1 while retaining other constraints, resulting in a new formation P' . Wherever ILP-1 improves upon ILP-0, such a reformulated P' may supposedly improve upon the original P .

This quality is not shared by other 1D-BPP reformulations as ILP such as are the state of the art today, since they depart further from the original encoding ILP-0, and also have a number of inequalities that is not polynomially bounded in the number of items.

4 Cutting Plane Constraints for the ILP-1 Formulation

We now introduce two families of constraints:

$$z_{jk} + z_{ij} \leq 1 \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, i - 1\}, k \in \{1, \dots, j - 1\} \quad (C1)$$

$$\sum_{k=1}^{j-1} z_{jk} + z_{ij} \leq 1 \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, i - 1\} \quad (C2)$$

We will now prove in several steps that these two sets of inequalities are actually cutting planes for the ILP-1 formulation.

Proposition 1. *The inequalities (C2) are valid for the ILP-1 formulation.*

Proof. Consider a feasible solution x for ILP-1, and let i in $\{1, \dots, n\}$, and k in $\{i + 1, \dots, n\}$. We distinguish two cases according to the value of z_{ij} .

- First we consider the case $z_{ij} = 0$. Then $\sum_{k=1}^{j-1} z_{jk} \leq \sum_{k=1}^j z_{jk} = 1$, as a consequence of constraint (8). So $\sum_{k=1}^{j-1} z_{jk} \leq 1$, and also $\sum_{k=1}^{j-1} z_{jk} + z_{ij} \leq 1$.
- In the case where $z_{ij} = 1$, we will first prove that $z_{jk} = 0$ for all $k < j$. Since $z_{ij} = 1$, the constraint (10) forces z_{jj} to equal 1. But then, according to constraint (9), all the z_{jk} for $k < j$ must equal 0. So $\sum_{k=1}^{j-1} z_{jk} = 0$, and

$$\text{hence } \sum_{k=1}^{j-1} z_{jk} + z_{ij} \leq 1. \quad \square$$

Proposition 2. *Let i be in $\{1, \dots, n\}$, j be in $\{1, \dots, i - 1\}$, and k be in $\{1, \dots, j - 1\}$. The inequality (C2) is stronger than the inequality (C1), when considered as reinforcements to the formulation ILP-1.*

Proof. In order to prove the statement, we will derive inequality (C1) from (C2). Assume (C2) holds, then:

$$z_{jk} + z_{ij} \leq \sum_{l=1}^{j-1} z_{jl} + z_{ij} \leq 1$$

where the first inequality holds, because in ILP-1, all variables are assumed to be positive. □

Proposition 3. *The inequality (C1) is a valid inequality for the ILP-1 formulation.*

Proof. This is the consequence of Propositions 1 and 2. □

Proposition 4. *There exist fractional solutions of ILP-1 that are separated by inequalities (C1) and (C2).*

Proof. We need to consider an instance of 1D-BPP such that there exists α with $1/2 < \alpha < 1$ and $w_1 + \alpha w_2 \leq C$.

Then the following is a feasible solution of the relaxation of ILP-1: set $z_{21} = z_{32} = \alpha$; $z_{22} = z_{33} = 1 - \alpha$; $z_{ii} = 1$ for all i in $\{1, 4, 5, \dots, n\}$; and $z_{ij} = 0$ for all other variables of the formulation.

This solution is represented in Fig. 3. Please note that the meaningful values all lie in the first three rows.

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & \dots & n \\
 \begin{array}{c}
 1 \\
 2 \\
 3 \\
 4 \\
 \vdots \\
 n
 \end{array}
 & \left(\begin{array}{cccccc}
 \color{red}{1} & \color{gray}{\times} & \color{gray}{\times} & \color{gray}{\times} & \dots & \color{gray}{\times} \\
 \color{red}{\alpha} & \color{red}{1-\alpha} & \color{gray}{\times} & \color{gray}{\times} & \dots & \color{gray}{\times} \\
 0 & \color{red}{\alpha} & \color{red}{1-\alpha} & \color{gray}{\times} & \dots & \color{gray}{\times} \\
 0 & 0 & 0 & \color{red}{1} & \dots & \color{gray}{\times} \\
 \vdots & \vdots & \vdots & \vdots & \ddots & \color{gray}{\times} \\
 0 & 0 & \dots & \dots & 0 & \color{red}{1}
 \end{array} \right)
 \end{array}
 \end{array}$$

Fig. 3. A feasible solution to the relaxation of ILP-1 that is cut by (C1)

Equations (8) are obviously satisfied by z . The inequality (8) holds because of the hypothesis we made for the case $j = 1$; because $(1 - \alpha)w_2 + \alpha w_3 \leq \max(w_2, w_3) \leq C$ for the case $j = 2$; and trivially holds for all the other cases.

But the inequalities (C1) will cut that point, namely z does not satisfy $z_{21} + z_{32} \leq 1$, since $\alpha + \alpha = 2\alpha > 1$.

Since the inequalities (C2) are stronger than (C1), these too are cuts for ILP-1. □

The number of inequalities in (C1) is $O(n^3)$, while it is $O(n^2)$ for (C2). Since (C2) inequalities are both much fewer and stronger than (C1), it seems that their use for performance improvement should be favored over (C1). In Section 5, we study empirically the performance benefits of adding one or the other of these families of inequalities.

5 Computational Experiments and Discussion

In this section, we analyze the performance of the new formulation ILP-1 without and with adding cutting planes. Our experiments were motivated by the following goals: comparing the number of optimally solved instances and the

run-time performances of our formulation ILP-1 with the standard formulation ILP-0 and determining the benefits obtained by including the cutting plane inequalities described in Sect. 4.

5.1 Setup

We implemented formulations ILP-0, ILP-1 and all its variants in Gurobi Optimizer 7.5.2 using Python 3.6 (<https://www.gurobi.com/>), running on a PC running Linux Debian 8.0 (“Jessy”). It has a Core 2 Duo CPU running at 3 GHz, and 4 GB of RAM. All executions were run within a single thread; only one core of the CPU was used.

5.2 Data-Sets

In order to test the performances of formulations ILP-0, ILP-1 and ILP-1+C k ($k \in \{1, 2\}$), we considered the data-sets from the literature of the 1D-BPP, referred to in the following as the BPPLIB and described in [4]. All instances are downloaded from the web page <http://or.dei.unibo.it/library/bpplib>. The main characteristics of the used data-sets are summarized in Table 3. Each data-set contains a number of tested instances (column #) of the 1D-BPP, characterized by having the same number of items (column n) and the same bin capacity (column C). Detailed information about the structure of each of these benchmarks can be found in [4] or in the BPPLIB web page.

Table 3. Main characteristics of the 9 used data-sets from the literature of the 1D-BPP (provided by the BPPLIB) considered in the experiments

Data-set	Ref.	Parameters of the instances		
		#inst.	n	C
Falkenauer T	[6]	40	{60, 120}	1000
Falkenauer U	[6]	40	{120, 250}	150
Scholl 1	[11]	360	{50, 100}	{100, 120, 150}
Scholl 2	[11]	240	{50, 100}	1000
Scholl 3	[11]	10	200	100 000
Schwerin 1	[12]	100	100	1000
Schwerin 2	[12]	100	120	1000
Wascher	[13]	17	[57 – 239]	10 000
Randomly generated	[3]	240	{50, 100}	{50, 75, 100, 120, 125, 150, 200, 300, 400, 500, 750, 1000}

5.3 Comparison of the ILP Models

In order to evaluate the formulations, ILP-0, ILP-1 and ILP-1+Ck for k in $\{1, 2\}$, we first compare its size complexity, which indicates how large a problem is in terms of binary variables and constraints as a function of n (the number of bins as well as of items). We note that in these formulations no Big-M constraints are considered.

The ILP-1 and ILP-1+Ck for k in $\{1, 2\}$ formulations have a smaller number of binary variables (n^2) than the ILP-0 ($n^2 + n$). On the other hand, both ILP-0 and ILP-1 are generally equivalent. They have the same order of number of constraints: $\mathcal{O}(n)$. In contrast, formulation ILP-1+C1 has the largest number of constraint with an order of $\mathcal{O}(n^3)$. Hence, the strengthening of formulation ILP-1 by cutting plane constraints seems to be more favorable for effectively reducing the search tree.

5.4 Computational Results: Analysis of the Gap and the Solution Times

In this section, we analyze our results under two main axes:

- **Axis 1: ILP-1 vs. ILP-0:** our goal is to assess the performance of the new symmetry-less ILP-1 against the standard ILP-0.
- **Axis 2: ILP-1+Ck for k in $\{1, 2\}$ vs. ILP-1:** our goal is to evaluate, with respect to ILP-1, the benefits obtained by including cutting plane inequalities.

Axis 1: ILP-1 vs. ILP-0. Table 4 gives the number of instances optimally solved in one CPU minute by the formulation ILP-0, respectively the formulation ILP-1. From these results, we can make the following observations:

- Formulation ILP-1 generally performs better than the formulation ILP-0, both when activating and deactivating the Gurobi Optimizer proprietary cuts. It was able to optimally solve within the time limit (60 s) in total 804 and 908 instances (in 5.7 and 4.5 s on average), respectively. Yet, the formulation ILP-0 was able to optimally solve within the time limit and in total only 597 and 584 instances (in 11 and 7.6 seconds on average), respectively.
- Formulation ILP-1 was able to solve within the time limit all the instances in the data-sets **FalkT**_(60,1000), **FalkT**_(120,150) and **Scho1**_(50,150), either when activating or deactivating Gurobi proprietary cuts (see Table 4). In contrast, the formulation ILP-0 was unable to solve any instance in the data-set **FalkT**_(60,1000) and able to optimally solve only 8 instances in the data-set **FalkT**_(120,150) and 47 instances in the data-set **Scho1**_(50,150) (in 30 or 16.8 and 1.7 or 1 s on average, respectively) when activating or deactivating the Gurobi proprietary cuts.
- Formulation ILP-0 provides the highest number of optimally solved instances only in both **Scho2**_(100,1000) and **Schw1**_(100,1000) data-sets (when

activating and deactivating the Gurobi proprietary cuts) and in the data-set **Schw2**_(120,1000) (when deactivating the Gurobi proprietary cuts). In addition, it was able to optimally solve a single instance in the data-set **Wae**_([57–239],10000).

Table 4. Number of instances solved in less than one minute (average CPU time in seconds), for formulations ILP-0 and ILP-1

Data-set	#inst.	ILP-0		ILP-1	
		No GC	With GC	No GC	With GC
FalkT _(60,1000)	20	0 (60.0)	0 (60.0)	20 (0.9)	20 (1.3)
FalkT _(120,1000)	20	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)
FalkU _(120,150)	20	8 (30.0)	8 (16.8)	16 (8.3)	20 (6.8)
FalkU _(250,150)	20	0 (60.0)	- (60.0)	9 (12.7)	- (60.0)
Scho1 _(50,100)	60	18 (1.3)	19 (0.8)	48 (0.2)	59 (0.04)
Scho1 _(50,120)	60	17 (2.2)	19 (2.9)	45 (0.1)	59 (0.1)
Scho1 _(50,150)	60	47 (1.7)	47 (1.0)	54 (0.7)	60 (0.1)
Scho1 _(100,100)	60	8 (12.9)	7 (4.3)	42 (0.3)	59 (0.3)
Scho1 _(100,120)	60	3 (15.0)	6 (23.4)	42 (2.8)	55 (1.4)
Scho1 _(100,150)	60	29 (11.5)	17 (11.3)	50 (6.7)	56 (2.7)
Scho2 _(50,1000)	120	111 (0.7)	113 (0.9)	112 (1.0)	118 (1.2)
Scho2 _(100,1000)	120	103 (1.5)	101 (1.7)	94 (3.0)	97 (4.6)
Scho3 _(200,100000)	10	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)
Schw1 _(100,1000)	100	52 (8.4)	48 (10.3)	39 (18.8)	32 (16.9)
Schw2 _(120,1000)	100	49 (8.8)	38 (9.8)	40 (20.3)	39 (20.9)
Wae _([57–239],10000)	10	1 (40.6)	- (60.0)	0 (60.0)	- (60.0)
RG	240	151 (8.1)	161 (8.1)	193 (4.4)	234 (2.0)
Total (average)	1140	597 (11.0)	584 (7.6)	804 (5.7)	908 (4.5)

Table 4 confirms the clear superiority of the formulation ILP-1 over the formulation ILP-0. This means that the new symmetry-less ILP for the 1D-BPP performs better.

Axis 2: ILP-1+C_k for k in $\{1, 2\}$ vs. ILP-1. Table 5 gives the number of instances solved in one CPU minute, by, respectively, the ILP-1 and ILP-1+C_k for k in $\{1, 2\}$ formulations. From these results, we can see that both formulations ILP-1+C_k for k in $\{1, 2\}$ generally have a similar performance, in particular they perform better than the formulation ILP-1, both when activating and deactivating the Gurobi Optimizer proprietary cuts. They were able to optimally solve within the time limit and when activating the Gurobi proprietary cuts in total 928 and 927 instances (in 6.5 and 5.1 s on average), respectively. Yet, the formulation ILP-1 was able to optimally solve within the time limit and in total only 908 instances in 4.5 s on average.

In contrast, the formulation ILP-1+C2 performs clearly better, when deactivating the Gurobi proprietary cuts, than the other two ILPs. In fact, it was able

to solve within the time limit in total 860 instances (in 5 s on average). Yet, both formulations ILP-1 and ILP-1+C1 were able to optimally solve within the time limit and in total only 804 and 835 instances (in 5.7 and 4.2 s on average), respectively. This means that the formulation ILP-1+C1 performs also better than the formulation ILP-1.

Table 5. Number of instances solved in less than one minute (average CPU time in seconds), for formulations ILP-1 and ILP-1+C k for k in $\{1, 2\}$

Data-set	#inst.	ILP-1		ILP-1+C1		ILP-1+C2	
		No GC	With GC	No GC	With GC	No GC	With GC
FalkT _(60,1000)	20	20 (0.9)	20 (1.3)	20 (2.0)	20 (2.7)	20 (1.9)	20 (2.5)
FalkT _(120,1000)	20	0 (60.0)	0 (60.0)	0 (60.0)	1 (37.9)	0 (60.0)	0 (60.0)
FalkU _(120,150)	20	16 (8.3)	20 (6.8)	19 (9.4)	20 (7.6)	18 (9.2)	20 (6.8)
FalkU _(250,150)	20	9 (12.7)	- (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)
Scho1 _(50,100)	60	48 (0.2)	59 (0.04)	54 (0.6)	60 (0.1)	55 (0.5)	60 (0.07)
Scho1 _(50,120)	60	45 (0.1)	59 (0.1)	50 (0.1)	60 (0.2)	54 (0.2)	60 (0.1)
Scho1 _(50,150)	60	54 (0.7)	60 (0.1)	56 (1.2)	60 (0.6)	57 (0.6)	60 (0.7)
Scho1 _(100,100)	60	42 (0.3)	59 (0.3)	45 (0.6)	59 (0.8)	47 (0.7)	59 (0.8)
Scho1 _(100,120)	60	42 (2.8)	55 (1.4)	43 (1.3)	57 (1.7)	45 (1.8)	56 (1.0)
Scho1 _(100,150)	60	50 (6.7)	56 (2.7)	50 (4.0)	57 (4.2)	52 (4.5)	55 (2.3)
Scho2 _(50,1000)	120	112 (1.0)	118 (1.2)	113 (1.0)	118 (0.4)	116 (1.6)	118 (0.6)
Scho2 _(100,1000)	120	94 (3.0)	97 (4.6)	96 (3.2)	99 (3.6)	95 (3.3)	104 (2.9)
Scho3 _(200,100000)	10	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)
Schw1 _(100,1000)	100	39 (18.8)	32 (16.9)	40 (12.4)	40 (13.2)	48 (18.4)	43 (21.7)
Schw2 _(120,1000)	100	40 (20.3)	39 (20.9)	36 (13.9)	40 (16.8)	37 (18.2)	34 (24.7)
Wae _(57-239,10000)	10	0 (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)
RG	240	193 (4.4)	234 (2.0)	213 (4.5)	237 (1.8)	216 (3.8)	238 (1.8)
Total (average)	1140	804 (5.7)	908 (4.5)	835 (4.2)	928 (6.5)	860 (5.0)	927 (5.1)

Table 5 clearly confirms the benefits obtained by including cutting plane inequalities to the new symmetry-less ILP formulation ILP-1. This means that formulations ILP-1+C k for k in $\{1, 2\}$ were slightly better than the new symmetry-less ILP formulation ILP-1.

6 Conclusion and Future Work

We have presented a study of how a new symmetry-less formulation can improve the resolution performance of Integer Linear Formulations for the 1-dimensional bin packing problem. Our study includes a folklore symmetry-less formulation and 2 series of cuts for this formulation. This folklore formulation encodes partitions directly, removing the need for variables to encode the use of bins.

One exciting perspective of this work would be to investigate the impact of reusing the concept of that folklore formulation, i.e. encoding partitions of sets, on the solution of other optimization problem ILP formulations, i.e. Bin Packing with Conflicts, Cutting Stock Problem, etc.

References

1. Arbib, C., Marinelli, F.: Maximum lateness minimization in one-dimensional bin packing. *Omega* **68**, 76–84 (2017)
2. Cambazard, H., O’Sullivan, B.: Propagating the bin packing constraint using linear programming. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 129–136. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15396-9_13
3. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.* **255**(1), 1–20 (2016)
4. Delorme, M., Iori, M., Martello, S.: BPPLIB: a library for bin packing and cutting stock problems. *Optim. Lett.* **12**(2), 235–250 (2018)
5. Dyckhoff, H.: A new linear programming approach to the cutting stock problem. *Oper. Res.* **29**(6), 1092–1104 (1981)
6. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. *J. Heuristics* **2**(1), 5–30 (1996)
7. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Program.* **100**(2), 423–445 (2004)
8. Martello, S., Toth, P.: Knapsack problems: algorithms and computer implementations (1990)
9. Martello, S., Toth, P.: Lower bounds and reduction procedures for the bin packing problem. *Discrete Appl. Math.* **28**(1), 59–70 (1990)
10. Ryan, D., Foster, E.: An integer programming approach to scheduling. Computer scheduling of public transport urban passenger vehicle and crew scheduling, pp. 269–280 (1981)
11. Scholl, A., Klein, R., Jürgens, C.: Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Comput. Oper. Res.* **24**(7), 627–645 (1997)
12. 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**(5–6), 377–389 (1997)
13. Wäscher, G., Gau, T.: Heuristics for the integer one-dimensional cutting stock problem: a computational study. *Oper.-Res.-Spectr.* **18**(3), 131–144 (1996)