An Experimental Study on Symmetry Breaking Constraints Impact for the One Dimensional Bin-Packing Problem

Khadija Hadj Salem, Yann Kieffer

Proceedings of the Federated Conference on Computer Science and Information Systems pp. 317–326
DOI: 10.15439/2020F19
ISSN 2300-5963 ACSIS, Vol. 21

An Experimental Study on Symmetry Breaking Constraints Impact for the One Dimensional Bin-Packing Problem

Khadija Hadj Salem
Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002,
64 avenue Jean Portalis, 37200 Tours
Email: khadija.hadj-salem@univ-tours.fr

Yann Kieffer
Univ. Grenoble Alpes, Grenoble INP, LCIS,
26000 Valence, France
Email: yann.kieffer@lcis.grenoble-inp.fr

***Abstract*—We consider the classical *One-Dimensional Bin Packing Problem* (1D-BPP), an $\mathcal{NP}$-hard optimization problem, where, a set of weighted items has to be packed into one or more identical capacitated bins. We give an experimental study on using symmetry breaking constraints for strengthening the classical integer linear programming proposed to optimally solve this problem. Our computational experiments are conducted on the data-sets found in BPPLib and the results have confirmed the theoretical results.**

I. Introduction

THE *one-dimensional Bin Packing Problem*, noted 1D-BPP from here on, has been widely studied in the literature both for its theoretical interest and its many practical applications. Several variants were considered as well as different approaches for its solution were proposed. The 1D-BPP 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 $(i = 1, \ldots, n)$ 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.

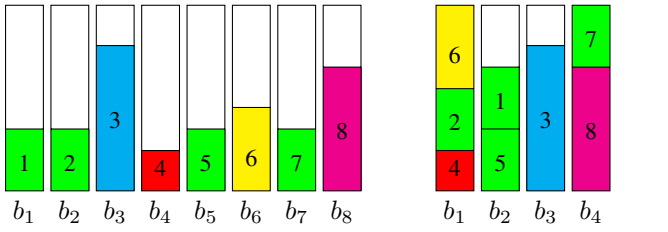
To illustrate these concepts, we consider the following example: give one instance of the 1D-BPP with a set of bins with capacity C equal to 6 and a set of items, indexed by i, with the weights w_i given in Table I.

TABLE I: 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

An example of a feasible solution as well as an optimal solution, respectively, with 8 bins and 4 bins, are given in Figure 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. So we conducted such a study, including a new inequalities to strengthen its basic mathematical model.



(a) A feasible solution: 8 bins **(b)** An optimal solution: 4 bins

Fig. 1: Solutions for the 1D-BPP

The remainder of this paper is organized as follows. Section II formally introduces the basic mathematical model and briefly mentions the exact solution methods considered in the literature of 1D-BPP. Section III gives a brief review on symmetries in ILP formulations. Sections IV — VII describe some classes of symmetry breaking constraints. Computational results are reported and analyzed in Section VIII. Finally, the main conclusions of this work as well as some future research directions are drawn.

II. Basic mathematical models for the 1D-BPP

A. Assignment-based models

The compact ILP formulation for 1D-BPP, which Martello and Toth attribute to Kantorovich (see [19]), is the following, by introducing two types of binary decision variables for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\}$.

- $y_j \begin{cases} 1 & \text{if bin } j \text{ is used in the packing} \\ 0 & \text{otherwise} \end{cases}$
- $x_{ij} \begin{cases} 1 & \text{if item } i \text{ is packed into bin } j \\ 0 & \text{otherwise} \end{cases}$

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

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

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

In this model, constraints (1) ensure that each item is packed into exactly one bin, constraints (2) impose that the capacity of any used bin is not exceeded and both constraints (3) and (4) 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`. Denoted by L_1 in the literature, this lower bound is given by the following equality:

$$\mathtt{L_1} = \left\lceil \sum_{j=1}^{n} w_i / C \right\rceil \quad (5)$$

It is easily seen that the *worst-case performance ratio* of L_1 is equal to $\frac{1}{2}$ (see, e.g., [20]).

The reader is referred to [11], which is a first survey on linear programming models for the 1D-BPP and its generalization, the Cutting Stock Problem (CSP).

B. Other methods for optimally solving the 1D-BPP

Among other methods for solving 1D-BPP exactly, we can find the branching algorithms and the pseudo-polynomial ILP formulations coming from a graph representation of the solution space.

The reader is referred to [15] for a recent survey on mathematical models and exact algorithms for both 1D-BPP and CSP and to [15] for a library, named BPPLIB and available at http://or.dei.unibo.it/library/bpplib. The BPPLIB provides a collection of computer codes of different types for the exact solution of the 1D-BPP and the CSP as well as a benchmark instance. It also includes a BibTeX file of more than 150 references on this topic and an interactive visual tool to manually solve both 1D-BPP and CSP.

An overview of these methods can be summarized as follows:

1) *Enumeration algorithms*, basically:
   - the branch-and-bound, in which three approaches are proposed: **MTP** (see [19]), **BISON** (see [1]) and **CVRPSEP** (see [9]).
   - the branch-and-price, in which one approach, called **SCIP-BP** (see [3]), is proposed.
2) *Pseudo-polynomial formulations solved through an ILP solver (like CPLEX, SCIP, GUROBI)*: we can find here both **ONECUT** (see [7]) and **DPFLOW** (see [6]).

According to the results discussed in [14], the **SCIP-BP** is effective on small-size instances ($n \leq 100$). In the same way, the **DPFLOW** has mainly theoretical interest, but has the advantage of being easily understandable.

III. A brief review on symmetries in ILP formulations

In a combinatorial optmization, symmetries increase the size of the search space and therefore, time to visiting symmetric solutions we will wasted. The most usual way to deal with symmetries is to add constraints that eliminate symmetric solutions. We give here a brief review on recent results in this area, focusing especially on the use of symmetry breaking constraints in mathematical programming models: LP [1], ILP [2] and MILP [3]. Please note that we refer here and after by the word ILP all variants of the mathematical programming models mentioned above.

Typical ILP formulations contain binary variables. An ILP is then symmetric if its variables can be permuted without changing the structure of the problem. For example, scheduling jobs on parallel identical machines or packing items into identical bins involve large symmetry groups.

For example, given a binary variable x_{ij}, where x_{ij} equal to one signifies that item i is assigned to bin j or that job i is assigned to machine j. The x variables can be interpreted as an $0-1$ matrix. Symmetry is often present in these kind of problems since there can be many identical bins/machines of a certain type. As result, given any feasible solution x, equivalent solutions can be generated by permuting the columns of x.

The presence of symmetry can have a significant negative effect on the performance of branch-and-bound algorithms. In the same way that it allows multiple equivalent solutions, symmetry also allows different sub-problems in the branch-and-bound tree to be equivalent.

First, Margot (2010) gives a survey of some of the approaches that have been developed for solving symmetric ILPs. These approaches are classified into four major groups: *perturbation*, *fixing variables*, *symmetry breaking inequalities*, and *pruning of the enumeration tree*. We refer to [5] for further details.

Similarly, Liberti (2012) gives a review of the most widespread approaches for breaking symmetries in ILPs together with a theorical and computational study of symmetries in the Kissing Number Problem (see [13]). In this paper, he used a generalization of the definition of formulation group given by Margot (2010), based on transforming an ILP into a DAG [4]. This allows automatic symmetry detection using graph isomorphism tools. Symmetries are then broken by means of static symmetry breaking inequalities.

In the same way, Sherali and Smith (2001) focus on the description of a natural method to remove symmetries in the context of the following problems: a *synchronous optical network (SONET) design problem*, a *minimax noise pollution problem*, and a *machine scheduling problem* (see [8]). Their method consists of augmenting the ILP model of that problem

[1] LP: Linear Programming
[2] ILP: Integer Linear Programming
[3] MILP: Mixed-Integer Linear Programming
[4] DAG: Directed Acyclic Graphs

with suitable symmetry breaking hierarchical constraints. The structure of the ILP can then be considerably improved by reducing the extent of the feasible region that must be explored by any algorithmic procedure.

Finally, Jans R. (2006) considers the issue of symmetry in the *lot-sizing problems on parallel identical machines* literature (see [18]). To break this symmetry, he simply enhances the existing ILPs by adding lexicographic ordering constraints. Other ways can be achieved by ordering the machines according to some natural logic (decreasing total setup cost per machine, decreasing total cost per machine or decreasing capacity utilization).

In summary, we can say that if symmetry is present in the ILP problem, it must be dealt with in an effective manner. There are many strategies that one can use to handle symmetries in the solution space. A most usual way is to add symmetry breaking constraints, as we can see in the next section in the case of 1D-BPP. In addition, an empirical evaluation of the impact of including separately or in combination the different symmetry breaking constraints to the `ILP-0` formulation is presented in Section VIII.

IV. A BASIC SYMMETRY BREAKING CONSTRAINT

Due to the fact that all bins $j \in \{1,\ldots,n\}$ are identical (the same integer capacity C), there is complete symmetry with respect to bins. Thus, for any solution, an equivalent solution can be obtained by swapping the sets of items assigned to any pair of bins. To break this symmetry and limit the number of mathematical solutions to the actual number of different allocations of bins, we first add the following constraints:

$$y_j \geq y_{j+1} \qquad \forall j \in \{1,\ldots,n-1\} \qquad \text{(S0)}$$

This constraint reduces the size of the enumeration tree by imposing that the bins are used in increasing order of index.

As we can see in Figures 2a and 2b, the optimal solution is defined by four bins. This means that the use of bins b_1, b_6, b_5, b_4 or bins b_1, b_2, b_3, b_4 is equivalent.

V. SORTING IN DECREASING ORDER OF BIN LOAD

The symmetry can be partially broken by stating that bins must be sorted by decreasing load. Consider the following constraint:

$$\sum_{i=1}^{n} w_i * x_{ij} \geq \sum_{i=1}^{n} w_i * x_{ij+1} \quad \forall j \in \{1,\ldots,n-1\} \quad \text{(S1)}$$

This constraint forces that the load of bin j must be greater than or equal to the load of bin $j+1$. An example of the effect of (S1) is given in Figure 3. The solution given in Figure 3a violates (S1), but the equivalent solution from Figure 3b respects it.

VI. SYMMETRY-LESS REFORMULATION: `ILP-0-S2+S0`

In this alternate formulation, the symmetry can be eliminated by stating that bins must be sorted by decreasing order of the maximum item index. To present this constraint, we define a new integer variable z_j ($j \in \{1,\ldots,n\}$) as the maximum index over all the items allocated to bin j.

In this case, the objective function is updated to the following weighted sum:

$$\min \sum_{j=1}^{n} y_j + \frac{2}{2+n(n+1)} * \sum_{j=1}^{n} z_j \qquad (6)$$

in a way that minimizes the number of used bins $\sum_{j=1}^{n} y_j$ first (primary objective) and then the sum of its maximum item index $\sum_{j=1}^{n} z_j$ (secondary objective). The latter is also weighted by the following coefficient $\frac{2}{2+n(n+1)}$ to impose the lexicographic optimization ordering as mentioned before. This means that $\sum_{j=1}^{n} y_j$ is the integer part of the objective function and that $\frac{2}{2+n(n+1)} * \sum_{j=1}^{n} z_j < 1$. Indeed:

$$\sum_{j=1}^{n} z_j = \frac{n(n+1)}{2} < \frac{n(n+1)}{2} + 1 = \frac{2+n(n+1)}{2}$$

$$\frac{n(n+1)}{2} < \frac{2+n(n+1)}{2}$$

$$\text{hence} \quad \frac{2}{2+n(n+1)} \sum_{j=1}^{n} z_j < 1$$

The constraints of `ILP-0-S2+S0` formulation are constraints (1)—(4) and the following set of inequalities which is denoted as (S2):

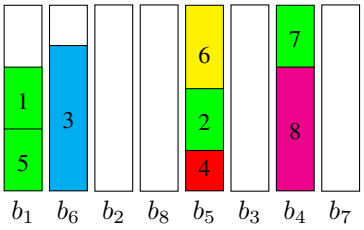
$$\begin{cases} i * x_{ij} \leq z_j & \forall\, i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,n\} \quad (7) \\ z_j \geq z_{j+1} & \forall\, j \in \{1,\ldots,n-1\} \quad (8) \\ z_j \geq 0 & \forall\, j \in \{1,\ldots,n\} \quad (9) \end{cases}$$

Constraints (7) ensure that the bin index z_j must be greater than or equal to the the maximum item index in bin j. Constraints (8) mean that the bin index bin z_j must be greater than or equal to the maximum item index in bin $j+1$. Constraints (8) guarantee the positivity of the bin index z_j. In addition, we consider the inequality constraint (S0) to reduce the size of the enumeration tree by imposing that the bins are used in increasing order of index.

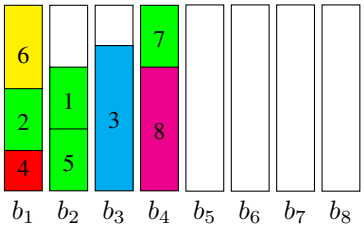
As illustrated in Figure 4b, the corresponding solution provided by this new formulation `ILP-0-S2+S0` is equivalent to the initial one given in Figure 4a.

VII. MATRIX-BASED SYMMETRY BREAKING CONSTRAINTS

The matrix-based symmetry breaking constraints, proposed here, were inspired from those proposed by [2] and reused by [10] for the classical *Job Scheduling Problem* (specifically operating room scheduling problems), which is a well known

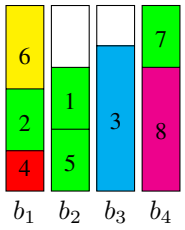


(a) A solution without using (S0)

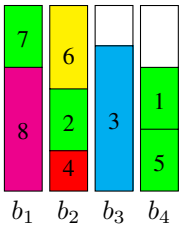


(b) A solution using (S0)

Fig. 2: Illustration of (S0)

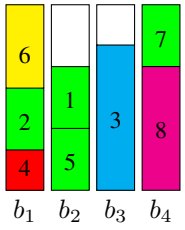


(a) A solution without using (S1)

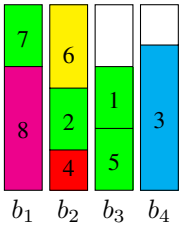


(b) A solution using (S1)

Fig. 3: Illustration of (S1)



(a) A solution without using S_2



(b) A solution using S_2

Fig. 4: Illustration of `ILP-0-S2+S0`

$\mathcal{NP}$-hard combinatorial optimization problem. These constraints restrict the feasible region to a minimal fundamental domain that has lexicographically decreasing columns.

For example, as illustrated in Figure 5, the 0/1 matrix x does not have lexicographically decreasing columns, so it is not in the fundamental domain. Permuting columns 1 and 2 gives the 0/1 matrix x', which has lexicographically-decreasing columns, so it is in the fundamental domain.

In the context of 1D-BPP, we introduce the following inequality (S3), which guarantees that for any x_{ij} equal to one with i and j greater than one, there is at least one lower-indexed item i assigned to bin $j-1$.

$$x_{ij} \leq \sum_{p=1}^{i-1} x_{p,j-1} \quad \forall i \in \{2,\ldots,n\}, \forall j \in \{2,\ldots,n\} \quad \text{(S3)}$$

Now, as an alternative to asking for a lower-indexed item in the previous bin, one could as well ask for a bigger-indexed item in the previous bin. This condition is expressed by the following constraint:

$$x_{ij} \leq \sum_{p=i}^{n-1} x_{p,j-1} \quad \forall i \in \{2,\ldots,n\}, \forall j \in \{i,\ldots,n\} \quad \text{(S3Bis)}$$

Of course, only one of (S3) or (S3Bis) can be enforced, since the two constraints are incompatible.

While the only $0-1$ matrices that satisfy constraints (S3) have lexicographically decreasing columns, the constraints can be strengthened to create a tighter LP relaxation.
Using the property that each row of x must contain a single one, the general form of constraints in (S3) becomes:

$$\sum_{s=j}^{n} x_{is} \leq \sum_{p=1}^{i-1} x_{p,j-1} \quad \forall\ i \in \{2,\ldots,n\}, \forall j \in \{2,\ldots,n\} \quad \text{(S4)}$$

In the same way, the general form of constraints in (S3Bis) becomes:

$$\sum_{s=j}^{n} x_{is} \leq \sum_{p=i}^{n-1} x_{p,j-1} \quad \forall\ i \in \{2,\ldots,n\}, \forall j \in \{2,\ldots,n\} \quad \text{(S4Bis)}$$

VIII. Computational Experiments and Discussion

In this section, we test the effectiveness of the formulations described in the previous sections. Our experiments were motivated by this main goal: to evaluate, with respect to the standard formulation `ILP-0` (see Section II-A), the benefits obtained by including the valid inequalities of breaking symmetries previously described in Sections IV — VII.

A. Setup

We implemented formulations `ILP-0` 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 with 4 gigabytes of RAM. All executions where run within a single thread; only one core of the CPU was used.

We considered seven variants of the formulation `ILP-0`:

- one is the classical ILP model of 1D-BPP: `ILP-0` (see Section II-A);
- one including the symmetry breaking constraint, given by Eq. (S0), hereafter denoted as `ILP-0+S0` (see Section IV);

$$x = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

(a) 0/1 matrix x

$$x' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

(b) 0/1 matrix x'

Fig. 5: Illustration of S_3

- one is the new alternate symmetry ILP model of 1D-BPP: `ILP-0-S2+S0` (see Section VI);
- one including the symmetry breaking constraints, given by Eq. (S0) and Eq. (S3), hereafter denoted as `ILP-0+S0+S3` (see Section VII);
- one including the symmetry breaking constraints, given by Eq. (S0) and Eq. (S3Bis), hereafter denoted as `ILP-0+S0+S3Bis` (see Section VII);
- one including the symmetry breaking constraints, given by Eq. (S0) and Eq. (S4), hereafter denoted as `ILP-0+S0+S4` (see Section VII);
- one including the symmetry breaking constraints, given by Eq. (S0) and Eq. (S4Bis), hereafter denoted as `ILP-0+S0+S4Bis` (see Section VII).

Moreover, for each implementation we considered two kinds of run-time settings. Specifically, we run each formulation by activating and deactivating Gurobi proprietary cuts, aiming at empirically validating the theoretical results presented in the previous sections.
As a Gurobi settings and in each implementation, we activated both Gurobi heuristics (with its default value of 0.05) and the presolving strategies and deactivated the Gurobi proprietary symmetry. In addition, in implementations `ILP-0+S0+S3`, `ILP-0+S0+S3Bis`, `ILP-0+S0+S4` and `ILP-0+S0+S4Bis`, we set the different added valid inequalities (symmetry breaking or cutting plane) as lazy inequalities, with a value of 2, i.e., all lazy constraints that are violated by a feasible solution will be pulled into the model. In contrast, we used a particular branching strategy for binary variables consisting of giving priority to x variables with respect to y variables during the branching process. We then set the ordering branch variable value to 1.

B. Data-sets

To test the performances of formulations `ILP-0`, `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` ($i \in \{3, 3Bis, 4, 4Bis\}$), we considered the data-sets from the literature of the 1D-BPP, referred to in the following as the `BPPLIB` and described in [14]. 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 II. Each data-set contains a number of tested instances (column **Tested inst.**) of the 1D-BPP, ad-hoc or uniformly distributed (column **Distribution**), 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 [15] or in the the BPPLIB web page.

C. Comparison of the ILP Models

To evaluate the different proposed ILP formulations, described in a previous sections, 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 integer variables, except for the formulation `ILP-0-S2+S0` (n integer variables) as well as no Big-M constraints are considered. As we can see in Table III, the `ILP-0` and its variants are generally equivalent in terms of binary variables. On the other hand, we can see that the three formulations: `ILP-0`, `ILP-0+S0`, `ILP-0+S0+S1` have the same order of number of constraints: $\mathcal{O}(n)$. In the same way, formulations `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, 4Bis\}$ have approximately the same order of $\mathcal{O}(n^2)$ of constraints number. Hence, the strengthening of the `ILP-0` by symmetry breaking inequalities seems to be more favorable for effectively reducing the search tree.

D. Numerical results

In this section, we analyze our results under one main axis, in which `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, S4Bis\}$ vs. `ILP-0` are compared. Our goal is to evaluate, with respect to `ILP-0`, the benefits obtained by including symmetry breaking constraints.

1) Analysis of the solution times: Table IV provides the results for the literature instances (described in table II) obtained by running the ILP formulations with a time limit of 60 seconds. Columns 1 and 2 identify the benchmarks (characterized by a specific number of items n and a bin capacity C) and give the number of instances for which the ILP formulations were executed. The column associated with each ILP formulation provides the number of such instances that were solved to proven optimality and, in parentheses, the average CPU time in the following two cases: `With GC` and

TABLE II: 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	Reference	Parameters of the instances			
		Tested inst.	n	C	Distribution
Falkenauer T	[4]	40	$\{60, 120\}$	1000	ad-hoc
Falkenauer U	[4]	40	$\{120, 250\}$	150	uniform
Scholl 1	[1]	360	$\{50, 100\}$	$\{100, 120, 150\}$	uniform
Scholl 2	[1]	240	$\{50, 100\}$	1000	uniform
Scholl 3	[1]	10	200	100 000	uniform
Schwerin 1	[17]	100	100	1000	uniform
Schwerin 2	[17]	100	120	1000	uniform
Wascher	[21]	17	$[57 - 239]$	10 000	ad-hoc
Randomly Generated	[14]	240	$\{50, 100\}$	$\{50, 75, 100, 120, 125, 150, 200, 300, 400, 500, 750, 1000\}$	ad-hoc

TABLE III: Comparison of ILP formulations

Models	No. variables		No. Constraints
	binary	integer	
ILP-0	$n^2 + n$	0	$2n$
ILP-0+S0	$n^2 + n$	0	$3n - 1$
ILP-0+S0+S1	$n^2 + n$	0	$4n - 1$
ILP-0-S2+S0	$n^2 + n$	n	$n^2 + 5n - 2$
ILP-0+S0+S3	$n^2 + n$	0	$(n^2)/2 + 3n/2$
ILP-0+S0+S3Bis	$n^2 + n$	0	$(n^2)/2 + 7n/2 - 1$
ILP-0+S0+S4	$n^2 + n$	0	$(n^2)/2 + 7n/2 - 1$
ILP-0+S0+S4Bis	$n^2 + n$	0	$(n^2)/2 + 7n/2 - 1$

No GC which refer to the activation and the deactivation of Gurobi proprietary cuts, respectively. For instances not solved, the time limit is considered as the solution time. A cell with a value of $-$ means that no feasible solution found when solving an instance using Gurobi optimizer within the time limit. For each instance set, **boldface** highlights the highest number of instances optimally solved. In the same way, for each instance set, colored cell (Blue) highlights the cases where all instances were solved to proven optimality. Finally, row **Total (average)** reports the total number of instances optimally solved within the time limit for each formulation as well as the average CPU time in seconds for its resolution.

The results in Table IV provide the number of instances solved in less than one minute (average CPU time in seconds), by, respectively, ILP-0, ILP-0+S0, ILP-0-S2+S0 and ILP-0+S0+Si for i in $\{3, 3Bis, 4, S4Bis\}$.

The results showed that all formulations were unable to solve any instance in the data-sets **FalkT**$_{(60,1000)}$, **FalkT**$_{(120,1000)}$, **FalkU**$_{(250,150)}$ and **Scho3**$_{(200,100000)}$ within the time limit, either when activating or deactivating the Gurobi Optimizer proprietary cuts, expect in the case of formulation ILP-0+S0+S4Bis, i.e., it was able to optimally solve 19 instances in less than 20 seconds on average either both cases respect to the proprietary cuts.

This trend is also marked in the case of the data-set **Wae**$_{([57-239],10000)}$ when activating the Gurobi Optimizer proprietary cuts, expect in the case of both ILP-0+S0+S4 and ILP-0+S0+S4Bis formulations, i.e., they were able to optimally solve only one instance in less than 50 seconds.

Formulation ILP-0 was able to solve within the time limit only 8 instances in the data-set **FalkU**$_{(120,150)}$, both when activating and deactivating the Gurobi Optimizer proprietary cuts. Unfortunately, the combination of formulation ILP-0 with the symmetry breaking constraints (ILP-0+S0, ILP-0-S2+S0 and ILP-0+S0+S3Bis) did not prove successful, i.e., only one or at most two instances can be optimally solved. In contrast, formulation ILP-0+S4Bis was able to optimally solve the biggest number of instances (11 instances in 26.8 seconds on average) when activating the Gurobi Optimizer proprietary cuts.

Formulations ILP-0, ILP-0+S0 ,ILP-0-S2+S0 and ILP-0+S0+S3Bis generally have a similar performance, i.e., they give rise to too similar results in the data-sets **Scho1**$_{(50,C)}$, for C in $\{100, 120, 150\}$ in both cases regarding the cuts proprietary. However, formulation ILP-0+S4Bis performs clearly better than the other formulations by giving rise to the best results in the data-sets **Scho1**$_{(50,C)}$ for C in $\{100, 120, 150\}$, i.e., it was able to solve within the time limit all the instances when activating the Gurobi Optimizer proprietary cuts, expect in the case of the data-sets **Scho1**$_{(50,C)}$ for C in $\{120, 150\}$ when deactivating the Gurobi Optimizer proprietary cuts.

Formulation ILP-0+S0+S4Bis provides the highest number of optimally solved instances in the data-sets **Scho1**$_{(100,C)}$, for C in $\{100, 120, 150\}$ when deactivating and activating the Gurobi Optimizer proprietary cuts, i.e., it was able to solve more than 40 instances in each case. In other hand, the behavior of both formulations ILP-0+S3 and ILP-0+S4 was similar. These formulations were unable to solve no instances, expect 3 instances in the data-set **Scho1**$_{(100,120)}$ were solved by ILP-0+S4 when activating the Gurobi Optimizer proprietary cuts.

Formulations ILP-0, ILP-0+S0 and ILP-0-S2+S0 give rise to the same results in the data-set **Scho2**$_{(50,1000)}$ when deactivating the Gurobi Optimizer proprietary cuts, i.e., they were able to optimally solve the highest number of instances (111 instances in less than 1 second on average). However, when activating the Gurobi Optimizer proprietary cuts, formulation ILP-0 provides the highest number of

TABLE IV: Number of instances solved in less than one minute (average CPU time in seconds), for formulations ILP−0, ILP−0+S0, ILP−0−S2+S0 & ILP−0+S0+Si for i in $\{3, 3Bis, 4, 4Bis\}$

Set	Tested inst.	ILP−0		ILP−0+S0		ILP−0−S2+S0		ILP−0+S3		ILP−0+S3Bis		ILP−0+S4		ILP−0+S4Bis	
		No GC	With GC	No GC	With GC	No GC	With GC	No GC	With GC	No GC	With GC	No GC	With GC	No GC	With GC
FalkT$_{(60,1000)}$	20	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	- (60.0)	- (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	**19** (19.8)	**19** (15.1)
FalkT$_{(120,1000)}$	20	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	- (60.0)	- (60.0)	0 (60.0)	0 (60.0)	- (60.0)	- (60.0)	0 (60.0)	0 (60.0)
FalkU$_{(120,150)}$	20	**8** (30.0)	8 (16.8)	1 (1.1)	2 (1.7)	2 (28.6)	1 (38.5)	- (60.0)	- (60.0)	2 (24.9)	2 (37.3)	- (60.0)	- (60.0)	6 (26.9)	**11** (26.8)
FalkU$_{(250,150)}$	20	0 (60.0)	- (60.0)	0 (60.0)	- (60.0)	0 (60.0)	- (60.0)	- (60.0)	- (60.0)	0 (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)
Scho1$_{(50,100)}$	60	18 (1.3)	19 (0.8)	16 (6.8)	20 (3.9)	16 (2.8)	20 (2.1)	- (60.0)	- (60.0)	18 (2.8)	16 (0.4)	3 (30.0)	37 (14.5)	**60** (0.3)	**60** (0.2)
Scho1$_{(50,120)}$	60	17 (2.2)	19 (2.9)	13 (9.1)	26 (2.7)	11 (3.1)	26 (3.6)	2 (46.1)	2 (27.3)	17 (4.2)	22 (1.7)	6 (7.3)	38 (10.0)	**57** (0.6)	**60** (0.4)
Scho1$_{(50,150)}$	60	47 (1.7)	47 (1.0)	40 (4.6)	43 (2.4)	40 (3.1)	44 (1.7)	6 (21.8)	6 (33.5)	47 (2.7)	48 (2.8)	16 (14.7)	18 (15.0)	**56** (0.9)	**60** (1.3)
Scho1$_{(100,100)}$	60	8 (12.9)	7 (4.3)	4 (12.2)	11 (13.0)	4 (12.7)	10 (7.5)	- (60.0)	- (60.0)	6 (9.9)	8 (9.9)	- (60.0)	- (60.0)	**54** (2.8)	**60** (2.3)
Scho1$_{(100,120)}$	60	3 (15.0)	6 (23.4)	1 (1.0)	8 (9.5)	0 (60.0)	8 (10.9)	- (60.0)	- (60.0)	6 (3.6)	6 (14.8)	- (60.0)	3 (36.9)	**51** (4.4)	**59** (5.0)
Scho1$_{(100,150)}$	60	29 (11.5)	29 (11.3)	17 (16.5)	17 (9.4)	17 (9.7)	17 (3.5)	- (60.0)	- (60.0)	21 (11.5)	26 (10.4)	0 (60.0)	0 (60.0)	**42** (13.6)	**43** (11.6)
Scho2$_{(50,1000)}$	120	**111** (0.7)	**113** (0.9)	**111** (0.3)	111 (0.3)	**111** (0.3)	111 (0.6)	97 (5.6)	98 (7.5)	110 (0.5)	111 (0.7)	108 (2.2)	110 (2.9)	107 (2.2)	111 (2.6)
Scho2$_{(100,1000)}$	120	**103** (1.5)	101 (1.7)	102 (2.5)	**102** (1.8)	101 (3.3)	101 (1.7)	46 (27.6)	44 (23.7)	92 (4.2)	98 (3.9)	70 (15.7)	72 (15.5)	57 (11.6)	57 (13.4)
Scho3$_{(200,100000)}$	10	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	0 (60.0)	- (60.0)	- (60.0)	0 (60.0)	0 (60.0)	- (60.0)	- (60.0)	- (60.0)	- (60.0)
Schw1$_{(100,1000)}$	100	52 (8.4)	48 (10.3)	40 (6.3)	52 (4.7)	40 (6.5)	51 (5.0)	- (60.0)	- (60.0)	**56** (13.4)	**62** (12.1)	12 (24.7)	11 (22.2)	15 (18.4)	10 (26.0)
Schw2$_{(120,1000)}$	100	**49** (8.8)	38 (9.8)	26 (8.4)	43 (10.8)	28 (10.0)	43 (11.8)	- (60.0)	- (60.0)	44 (13.2)	**46** (10.8)	2 (37.6)	1 (23.0)	9 (20.6)	6 (24.2)
Wae$_{([57-239],10000)}$	10	1 (40.6)	- (60.0)	8 (15.5)	- (60.0)	**9** (17.8)	- (60.0)	- (60.0)	- (60.0)	1 (37.2)	- (60.0)	2 (35.9)	**1** (42.3)	- (60.0)	**1** (49.8)
RG	240	151 (8.1)	161 (8.1)	103 (7.9)	117 (6.3)	97 (5.4)	108 (6.6)	3 (35.0)	44 (20.3)	140 (8.2)	153 (10.6)	16 (21.4)	22 (22.8)	**189** (4.5)	**212** (5.2)
Total (average)	1140	597 (11.0)	596 (7.6)	482 (7.1)	552 (5.6)	476 (8.6)	540 (7.8)	154 (27.2)	194 (22.4)	560 (10.5)	598 (9.6)	235 (21.0)	314 (20.51)	**722** (9.7)	**769** (13.1)

optimally solved instances, with a value of 113 (in less than 1 second on average).

In the case of the data-set **Scho2**$_{(100,1000)}$, formulation ILP−0 performs better than the other formulations when deactivating the Gurobi Optimizer proprietary cuts, i.e., it was able to optimally solve 103 instances. In contrast, when activating the Gurobi Optimizer proprietary cuts, formulation ILP−0−S2+S0 is better by optimally solving 102 instances in less than 2 seconds on average.

In both **Schw1**$_{(100,1000)}$ and **Schw2**$_{(120,1000)}$ data-sets, the table shows the clear superiority of formulation ILP−0+S0+S3Bis over the other formulations, either when activating or deactivating the Gurobi Optimizer proprietary cuts, expect in the case of the data-set **Schw2**$_{(120,1000)}$, for which formulation ILP−0 performs better when activating Gurobi Optimizer proprietary cuts.

Among the proposed symmetry breaking constraints, formulation ILP−0+S0+S4Bis provides the highest number of optimally solved instances in the data-set **RG** (Randomly Generated) compared to the formulation ILP−0. It was able to solve 189 and 212 instances (in 5 seconds on average), when activating and deactivating the Gurobi Optimizer proprietary cuts, respectively.

The table confirms the clear superiority of ILP−0+S0+S4Bis over the other formulations. It was able to optimally solve within the time limit (60 seconds) in total 722 and 769 instances (in 9.7 and 13.1 seconds on average), respectively. In the same way, we can see that the behavior of ILP−0+S0, ILP−0−S2+S0, ILP−0+S0+S3Bis and ILP−0+S0+S3Bis formulations was generally similar to that of the standard formulation ILP−0. In addition, we can see that the formulation ILP−0+S0+S4, where constraints (S4) is the general form of constraints in (S3), performs better than formulation ILP−0+S0+S3, especially when activating the Gurobi Optimizer proprietary cuts. For example, formulation ILP−0+S0+S3 was unable to solve within the time limit any instance in the data-sets **FalkU**$_{(120,150)}$, **Scho1**$_{(50,100)}$, **Scho2**$_{(100,C)}$ for C in $\{100, 120, 150\}$, **Schw1**$_{(100,1000)}$, **Schw2**$_{(120,1000)}$ and **Wae**$_{([57-239],10000)}$. This means that the constraint (S3) is the least effective among all the other symmetry breaking constraints.

However, the classical formulation ILP−0 remains an effective model to solve some of the used BPPLIB data-sets, specifically the data-sets **Scho2**$_{(n,100)}$ for n in $\{50, 100\}$ and **Schw2**$_{(120,1000)}$.

As a general trend, the results showed that the use of Gurobi Optimizer proprietary cuts may increase the number of instances optimally solved and reduce the solution time in most formulations with the exception of the formulation ILP−0 for certain data-sets: **Scho1**$_{(100,100)}$, **Scho1**$_{(100,150)}$, **Scho2**$_{(n,100)}$ for n in $\{50, 100\}$, **Schw2**$_{(100,1000)}$ and **Schw2**$_{(120,1000)}$.

2) Analysis of the gap : We analyze the performances of each proposed formulation with respect to both Gurobi gap GGap (%), i.e., the difference between the best feasible solution and the best lower bound found by Gurobi at the end of CPU time limit (60 s) and the Gap, i.e., the difference between the best lower bound to a given instance of the 1D-BPP in a specific data-set and the objective function value of the linear programming relaxation at the root node of the respective search tree, divided by the best lower bound.

Figs. 6, 8, 7 and 9 show the results of our computational experiments in term of box-and-whisker plots. Specifically, the bottom and the top of each box represent the first and third quartiles; the band inside the box represents the second quartile (the median), and the ends of the whiskers represent the 9th percentile and the 91st percentile. Outliers are plotted as individual points.

In particular, Figs. 6 and 7 show the performances of the 7 formulations when disabling Gurobi Optimizer proprietary cuts. In contrast, Figs. 8 and 9 show the performances of the 7 formulations when enabling Gurobi Optimizer proprietary cuts. The gaps are expressed in percentage and the performances are (i) represented by means of box and whiskers plots and (ii) shown in function of 9 data-sets: **FalkU**$_{(120,150)}$,

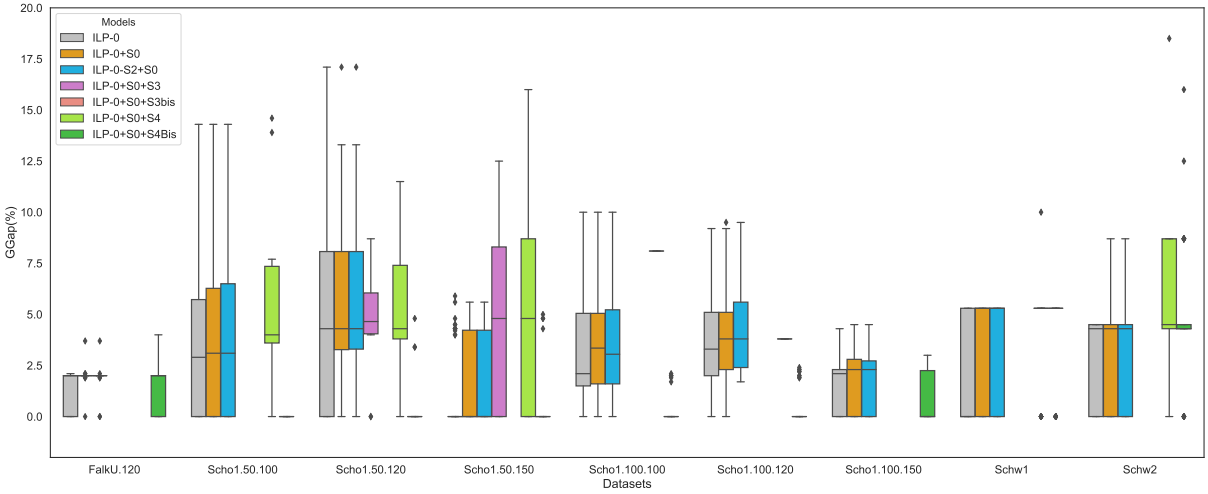


Fig. 6: Comparison of the gurobi gaps (%) of formulations `ILP-0`, `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, 4Bis\}$ on the data-sets from BPPLIB, when disabling Gurobi Optimizer proprietary cuts

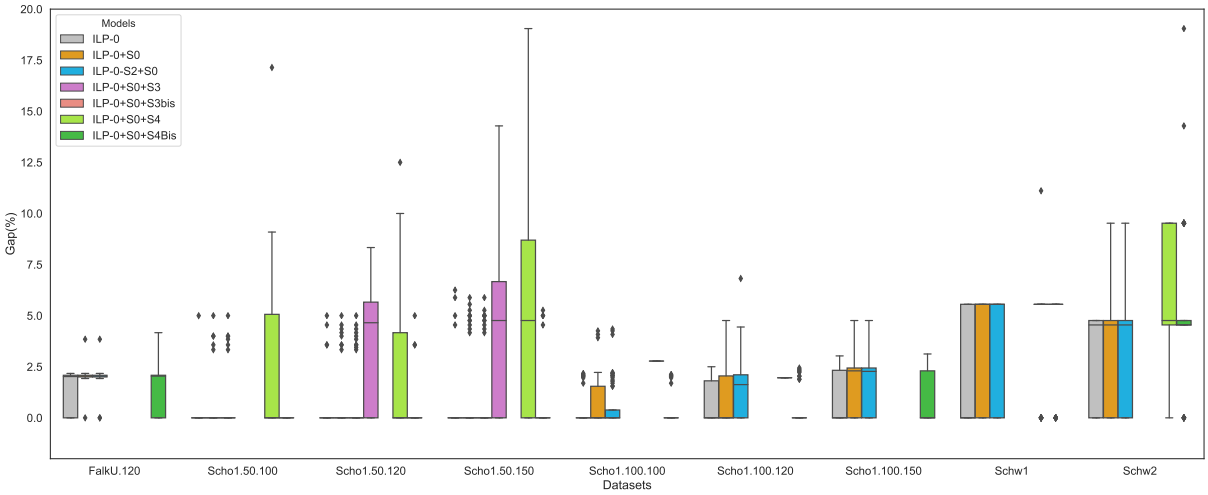


Fig. 7: Comparison of the gaps (%) of formulations `ILP-0`, `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, 4Bis\}$ on the data-sets from BPPLIB, when disabling Gurobi Optimizer proprietary cuts

Scho1$_{(50,C)}$, for C in $\{100, 120, 150\}$), **Scho1**$_{(100,C)}$, for C in $\{100, 120, 150\}$), **Schw1**$_{(100,1000)}$ and **Schw2**$_{(120,1000)}$. For the rest of data-sets, the behavior of all formulations was generally similar, as shown in Table IV.

As shown in Figs. 6 and 8, respectively in Figs. 7 and 9, formulation `ILP-0+S0+S4Bis` provides generally the smallest median gurobi gaps `GGap` and gaps `Gap`, except in the case of data-sets **FalkU**$_{(120,150)}$ and **Scho1**$_{(100,150)}$, in which its results are too similar to those of `ILP-0`. This fact is consistent with the results discussed in the previous sections.

In the same way, Figs. 6 and 8 show that the activation of the Gurobi Optimizer proprietary cuts from one hand causes a general decrement of the median gaps related to the formulations but on the other hand does not change their general trends. In particular, we observed that the use of these proprietary strategies has a major impact on formulations `ILP-0+S0`, `ILP-0-S2+S0`, `ILP-0+S0+S3` and `ILP-0+S0+S4`, minor in formulation `ILP-0` and becomes negligible or absent in formulations `ILP-0+S0+S3Bis` and `ILP-0+S0+S4Bis`. The trend is more marked in datasets **Scho1**$_{(50,C)}$, for C in $\{100, 120\}$) and **Scho1**$_{(100,C)}$, for C in $\{100, 120, 150\}$) and less in the others, for which the improvements are marginal.

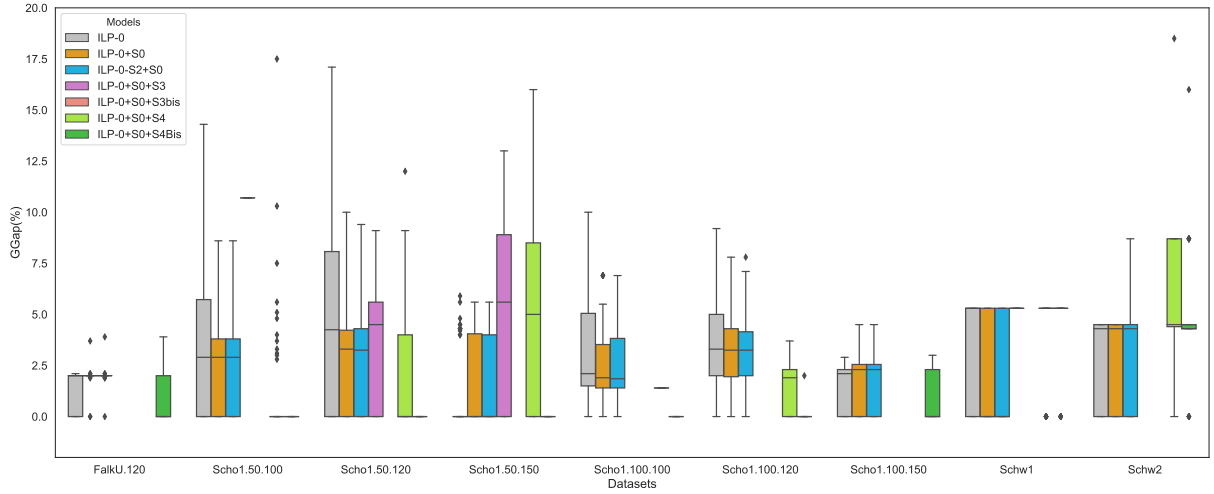


Fig. 8: Comparison of the gurobi gaps (%) of formulations `ILP-0`, `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, 4Bis\}$ on the data-sets from BPPLIB, when enabling Gurobi Optimizer proprietary cuts

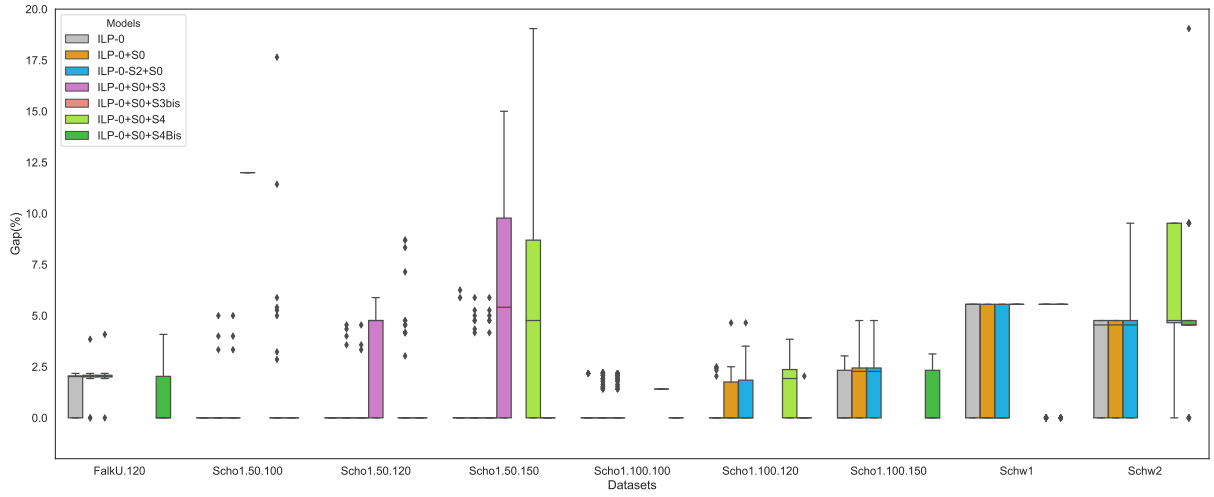


Fig. 9: Comparison of the gaps (%) of formulations `ILP-0`, `ILP-0+S0`, `ILP-0-S2+S0` and `ILP-0+S0+Si` for i in $\{3, 3Bis, 4, 4Bis\}$ on the data-sets from BPPLIB, when enabling Gurobi Optimizer proprietary cuts

Our 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 symmetry-less formulation for the 1D-BPP. This new formulation encodes partitions directly, removing the need for variables to encode the use of bins. We refer to Hadj Salem and Kieffer (2020) [12] for further details.

IX. Conclusion and Future Work

We have presented a study of how symmetry breaking constraints can improve the resolution performance of integer linear formulations for the 1-dimensional bin packing problem. Our study includes a review of all known and/or symmetry breaking constraints.

One exciting perspective of this work would be to investigate the impact of reusing these inequalities to other optimization problem ILP formulations, e.g., BPP with Conflicts (BPC), Cutting Stock Problem, etc. Understand how symmetry breaking methods interact with other ILP features such as branching strategies and cutting plane methods, in the context of packing problems, may be also a new direction should be investigated.

ACKNOWLEDGEMENTS

We are indebted to Professor André ROSSI (Paris-Dauphine University - LAMSADE) for suggesting both symmetry breaking constraints (S1) (see Section V) and the alternate formulation `ILP-0-S2+S0` (see Section VI), as well as for many constructive comments, that led to a clearer presentation.

REFERENCES

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

[2] B. T. Denton, A. J. Miller, H. J. Balasubramanian and T. R. Huschka, "Optimal allocation of surgery blocks to operating rooms under uncertainty," *Operations research,* vol. 58(4-part-1), 2010, pp. 802–816.

[3] D.M. Ryan and E.A. Foster, "An integer programming approach to scheduling," *Computer scheduling of public transport urban passenger vehicle and crew scheduling,* 1981, pp. 269–280.

[4] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics,* vol. 2(1), 1996, pp. 5–30.

[5] F. Margot, "Symmetry in integer linear programming," *50 Years of Integer Programming 1958-2008,* Springer, 2010, pp. 647–686.

[6] H. Cambazard and B. O'Sullivan, "Propagating the bin packing constraint using linear programming," *International Conference on Principles and Practice of Constraint Programming,* St. Andrews, Scotland, 2010, pp. 129–136.

[7] H. Dyckhoff, "A new linear programming approach to the cutting stock problem," *Operations Research,* vol. 29(6), 1981, pp. 1092–1104.

[8] H.D. Sherali and J.C. Smith, "Improving discrete model representations via symmetry considerations," *Management Science,* vol. 47(10), 2001, pp. 1396–1407.

[9] J. Lysgaard, A.N Letchford and R.W Eglese, "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming,* vol. 100(2), 2004, pp. 423–445.

[10] J. Ostrowski, M.F. Anjos and A. Vannelli, "Symmetry in scheduling problems," ,Citeseer, 2010, pp. 59–70.

[11] J. V. De Carvalho, "LP models for bin packing and cutting stock problems," *European Journal of Operational Research,* vol. 141(2), 2002, pp. 253-273.

[12] K. Hadj Salem and Y. Kieffer, "New Symmetry-less ILP Formulation for the Classical One Dimensional Bin-Packing Problem," 26^{th} *International Computing and Combinatorics Conference,* Atlanta, GA, USA, 2020, pp. 423-434.

[13] L. Liberti, "Symmetry in mathematical programming," *Mixed Integer Nonlinear Programming,* Springer, New York, NY, 2012, pp. 263–283.

[14] M. Delorme, M. Manuel and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *European Journal of Operational Research,* vol. 255, 2016, pp. 1–20.

[15] M. Delorme, M. Manuel and S. Martello, "BPPLIB: a library for bin packing and cutting stock problems," *Optimization Letters,* vol. 12(2), 2018, pp. 235–250.

[16] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi and L. A. Wolsey, "50 Years of integer programming 1958-2008: From the early years to the state-of-the-art," *Springer Science & Business Media,* 2009, pp. 123-136.

[17] P. Schwerin G. and Wäscher, "The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP," *International Transactions in Operational Research,* vol. 4(5-6), 1997, pp. 377–389.

[18] R. Jans, "Solving lot-sizing problems on parallel identical machines using symmetry breaking constraints," *INFORMS Journal on Computing,* vol. 21(1), 2009, pp. 123-136.

[19] S. Martello and P. Toth, "Knapsack problems: algorithms and computer implementations," *John Wiley & Sons, Inc.,* 1990, pp. 123-136.

[20] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete applied mathematics,* vol. 28(1), 1990, pp. 59–70.

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