The Bin Packing Problem with Setups: Formulation, Structural Properties and Computational Insights

Roberto Baldacci[a], Fabio Ciccarelli[b,*], Valerio Dose[b], Stefano Coniglio[c], Fabio Furini[b]

[a]*College of Science and Engineering Hamad Bin Khalifa University Qatar Foundation Doha Qatar*

[b]*Department of Computer Control and Management Engineering Antonio Ruberti Sapienza University of Rome Rome Italy*

[c]*Department of Economics. University of Bergamo Bergamo Italy*

Abstract

We introduce and study a novel generalization of the classical Bin Packing Problem (BPP), called the *Bin Packing Problem with Setups* (BPPS). In this problem, which has many practical applications in production planning and logistics, the items are partitioned into classes and, whenever an item from a given class is packed into a bin, a setup weight and cost are incurred. We present a natural *Integer Linear Programming* (ILP) formulation for the BPPS and analyze the structural properties of its *Linear Programming* relaxation. We show that the lower bound provided by the relaxation can be arbitrarily poor in the worst case. We introduce the *Minimum Classes Inequalities* (MCIs), which strengthen the relaxation and restore a worst-case performance guarantee of 1/2, matching that of the classical BPP. In addition, we derive the *Minimum Bins Inequality* (MBI) to further reinforce the relaxation, together with an upper bound on the number of bins in any optimal BPPS solution, which leads to a significant reduction in the number of variables and constraints of the ILP formulation. Finally, we establish a comprehensive benchmark of 480 BPPS instances and conduct extensive computational experiments. The results show that the integration of MCIs, the MBI, and the upper bound on the number of bins substantially improves the performance of the ILP formulation in terms of solution time and number of instances solved to optimality.

Keywords: Combinatorial Optimization, Bin Packing Problem, Integer Linear Programming, Computational Experiments

*Corresponding author. E-mail address: `f.ciccarelli@uniroma1.it`

1. Introduction

Given an infinite number of identical *bins* with a positive integer *capacity* $d \in \mathbb{Z}_{>0}$, and a set $\mathcal{I} = \{1, 2, \dots, n\}$ of n *items*, where each item $i \in \mathcal{I}$ has a positive integer *weight* $w_i \in \mathbb{Z}_{>0}$, the *Bin Packing Problem* (BPP) consists of determining a partition of the items into the minimum number of subsets such that the total weight of each subset does not exceed the bin capacity. This classical packing problem is fundamental in Operations Research and has been extensively studied from both theoretical and algorithmic perspectives; see Coffman et al. [13] and Delorme et al. [17] for comprehensive surveys.

In this paper, we introduce and study a generalization of the BPP, which we call the *Bin Packing Problem with Setups* (BPPS). In this new problem, the item set $\mathcal{I}$ is partitioned into m *item classes*, collectively denoted by $\mathcal{P} = \{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_m\}$, where, for each class $c \in \mathcal{C} = \{1, 2, \dots, m\}$, $\mathcal{I}_c \subseteq \mathcal{I}$ is the subset of items belonging to class c. Since the classes form a partition of $\mathcal{I}$, we have

$$\mathcal{I} = \bigcup_{c \in \mathcal{C}} \mathcal{I}_c \quad \text{and} \quad \mathcal{I}_c \cap \mathcal{I}_g = \emptyset \quad \text{for all } c, g \in \mathcal{C} \text{ with } c \neq g,$$

that is, each item belongs to exactly one class. For each class $c \in \mathcal{C}$, we are also given a nonnegative integer *setup weight* $s_c \in \mathbb{Z}_{\geq 0}$ and a nonnegative integer *setup cost* $f_c \in \mathbb{Z}_{\geq 0}$. Whenever at least one item of class c is packed into a bin, we incur both a setup cost f_c and a reduction in available bin capacity equal to the setup weight s_c. Each setup cost and weight is incurred only once per bin–class pair, regardless of how many items of that class are placed in the same bin. Throughout this paper, a class is said to be *active* in a bin if and only if the bin contains at least one of its items. Finally, we are given a positive integer *bin cost* $r \in \mathbb{Z}_{>0}$, which is incurred once for every used bin. We assume that all input data are integers, since any rational values can be scaled to integers without affecting the solution of the problem.

The BPPS consists of determining a minimum-cost partition of the items such that, in each subset(a bin), the total weight of its items plus the setup weights of all the classes that are active in it does not exceed the bin capacity. The cost of a partition is the sum, across all subsets, of the bin cost plus the setup costs of the classes that are active in it. We denote by ψ the optimal solution value of the BPPS, which corresponds to the minimum cost required to pack all items. An instance $I(BPPS)$ of the BPPS is defined by the tuple $(n, \boldsymbol{w}, d, m, \mathcal{P}, \boldsymbol{s}, \boldsymbol{f}, r)$, where $\boldsymbol{w} \in \mathbb{Z}^n_{\geq 0}$ is the vector of item weights, $\boldsymbol{s} \in \mathbb{Z}^m_{\geq 0}$ is the vector of setup weights, and $\boldsymbol{f} \in \mathbb{Z}^m_{\geq 0}$ is the vector of setup costs of the classes. To ensure feasibility of a BPPS instance, we assume that, for each class $c \in \mathcal{C}$, the combined weight of any item $i \in \mathcal{I}_c$ and its setup weight does not exceed the bin capacity, i.e., $w_i + s_c \leq d$. Moreover, we exclude the trivial case where all items can be packed in a single bin.

If $f_c = s_c = 0$ for all $c \in \mathcal{C}$, the BPPS is equivalent to the classical BPP. Since the BPP is strongly $\mathcal{NP}$-hard, the same complexity result applies to the BPPS. It is worth noticing that, when $r = 0$, the objective coincides with the total setup cost. In this case, the problem is equivalent to solving an instance of the BPP for each class $c \in \mathcal{C}$, where the capacity of the bin is reduced to $d - s_c$.

The bin cost r plays an important role in shaping the structure of optimal BPPS solutions. A high value of r discourages the use of multiple bins, promoting solutions that employ fewer bins packed as tightly as possible—even at the expense of activating multiple classes within the same bin, which increases the total setup cost. Conversely, when r is small, it becomes advantageous to use more bins to avoid incurring high setup costs from combining items of different classes in the same bin. This trade-off is illustrated by the two optimal BPPS solutions depicted in Figure 1.

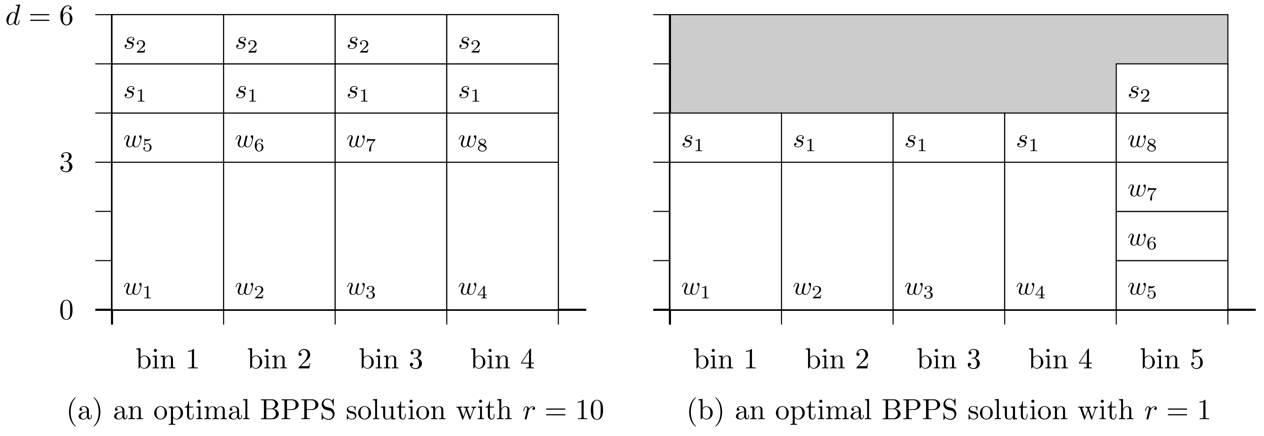

Figure 1: The figure considers a BPPS instance with bin capacity $d = 6$, number of items $n = 8$, and number of classes $m = 2$. The item weights are $w_1 = w_2 = w_3 = w_4 = 3$ and $w_5 = w_6 = w_7 = w_8 = 1$, while the setup weights and costs are $s_1 = s_2 = 1$, $f_1 = 2$ and $f_2 = 3$. The items are partitioned into the classes $\mathcal{I}_1 = \{1, 2, 3, 4\}$ and $\mathcal{I}_2 = \{5, 6, 7, 8\}$. The figure shows two optimal BPPS solutions for this instance for two different values of the bin cost r. Used bins are shown along the horizontal axis, while the vertical axis represents the bin capacity d. For each bin, the figure displays the total weight of the packed items and the setup weight associated with their classes. The height of each item and setup block corresponds to its weight, visually illustrating bin utilization. The remaining empty space in each bin is shown in gray. In part (a), with $r = 10$, the optimal solution consists of four used bins: $\big\{\{1, 5\}, \{2, 6\}, \{3, 7\}, \{4, 8\}\big\}$, yielding a total minimum cost of $\psi = 60$. In part (b), with $r = 1$, the optimal solution consists of five used bins: $\big\{\{1\}, \{2\}, \{3\}, \{4\}, \{5, 6, 7, 8\}\big\}$, with a total minimum cost of $\psi = 16$.

1.1. Applications and motivations

To the best of our knowledge, the BPPS has not yet been addressed in the literature, despite its high practical relevance. Setup costs and setup times are critical in many production planning applications where production lines or plants must be configured for specific activities; see, e.g., the books by Pochet and Wolsey [28] and Sawik [31] and the references therein. In such settings, products (items) are typically grouped into families (classes), and specific setup operations are required before manufacturing products of each family. A typical application of the BPPS in production planning can be described as follows. Consider a set of identical production lines, each available for a fixed amount of time (corresponding to the bin capacity), and a fixed cost incurred whenever a line is activated. A set of products must be manufactured, where each product requires a given processing time (item weight) and belongs to a single product class. Manufacturing items of a class on a production line requires a setup operation that consumes a predefined amount of time (setup weight) and incurs a monetary setup cost. These operations may represent, for example, the intervention of specialized technicians or the reconfiguration of equipment for a particular product family. In this context, the BPPS models the problem of determining a minimum-cost production plan. Consider, for instance, a production setting involving bottles, flasks, and pots, as in Chebil and Khemakhem [12], where machinery requires setup operations when switching from one product type to another or a series of sewing production lines, where garments of different shapes and colors are produced and color changes require specific setup actions; see, e.g., Focacci et al. [18].

Other important applications arise in logistics contexts where goods are distributed by vehicles requiring special handling equipment, such as forklifts or conveyors, or by multi-compartment vehicles that allow for transporting different types of goods in separate compartments [20, 29, 8]. In these applications, items represent customer orders to be delivered and are partitioned into classes, each requiring a specific piece of handling equipment. Bins correspond to the vehicles used for delivery. Assigning an order to a vehicle requires the corresponding equipment to be available on board, which leads to both a capacity reduction and an equipment (setup) cost—e.g., due to renting a forklift or installing specialized loading systems. Similarly, in the chemical and pharmaceutical industries [6], as well as in the distribution of perishable goods [8], certain products require refrigerated or frozen compartments. This reduces the available space in the vehicle (e.g., due to the installation of a bulkhead separating frozen and dry goods) and incurs additional costs (e.g., due to energy consumption for refrigeration). In such scenarios, the BPPS models the problem of determining the optimal composition of the fleet and assignment of customer orders to vehicles in order to minimize total distribution costs.

1.2. Literature review on related problems

The most closely related problem to the BPPS is the *Knapsack Problem with Setups* (KPS), introduced by Chajakis and Guignard [11], Lin [22]. In this problem, items are partitioned into classes, and a fixed setup cost is incurred for each class whose items are packed in the knapsack. The objective is to select a subset of items that maximizes the total profit, defined as the sum of item profits minus the setup costs of the activated classes. As in the BPPS, the setup associated with each active class also consumes part of the knapsack capacity, thereby creating a trade-off between item selection and class activation. Several exact and heuristic algorithms have been proposed for the KPS, including MILP formulations, decomposition-based methods, and various combinatorial approaches [1, 2, 12, 19, 14, 26, 27, 36]. Despite these similarities, the KPS differs fundamentally from the BPPS in that it involves a single knapsack rather than multiple bins, and does not require packing all items.

Our study also connects to the broader literature on generalizations of the classical Bin Packing Problem (BPP), aimed at capturing features that arise in complex real-world applications—see, e.g., [3, 10, 15, 16, 24, 30, 34]. Among the most relevant and related variants, the *Class-Constrained Bin Packing Problem* (CBPP) [7, 33, 35] assumes that items are partitioned into classes and imposes a limit on the number of distinct classes allowed per bin. The *Bin Packing Problem with Minimum Color Fragmentation* (BPPMCF) [4, 5, 9, 25] aims to minimize the number of bins used for each color, encouraging the grouping of items of the same type, under the constraint that only a fixed number of bins is available. While these models incorporate class-based constraints, they do not account for class-dependent setup costs or setup weights, nor do they model the trade-off between bin usage and class activation. To the best of our knowledge, the BPPS is the first problem to address this setting by considering that setup operations are both capacity-consuming and cost-inducing, depending on the classes of the items packed into each bin.

1.3. Contributions and structure of the paper

In Section 2, we propose a natural *Integer Linear Programming* (ILP) formulation for the BPPS, extending the classical formulation of the BPP. In Section 2.1, we study the structural properties of the *Linear Programming* (LP) relaxation of the ILP formulation, deriving closed-form expressions for its optimal solution and showing that the associated lower bound can be arbitrarily poor in the worst case. In Section 2.2, we introduce an effective family of valid inequalities, called the *Minimum Class Inequalities* (MCIs), which strengthen the LP relaxation of the ILP model. In Section 2.3, we analyze the structural properties of the LP relaxation with the MCIs, deriving closed-form expressions for its optimal solution. We then prove that incorporating the MCIs significantly improves the LP relaxation, guaranteeing a worst-case performance ratio of $1/2$ between the lower bound and the optimal BPPS solution value. In Section 2.4, we present an additional valid inequality, called the *Minimum Bins Inequality* (MBI), which further strengthens the LP relaxation of the ILP model. In Section 2.5, we derive closed-form expressions for the optimal solution of the relaxation with the MBI. Finally, in Section 2.6, we establish an upper bound on the number of bins used in any optimal BPPS solution. This result enables a significant reduction in the number of variables and constraints of the ILP formulation. Most of the proofs of the theoretical results are provided in the electronic companion due to space limitations. Section 3 presents the computational study. Since the BPPS is a novel problem, we first introduce in Section 3.1 a benchmark library of synthetic instances that covers its main features, including variations in item weights, class numbers, setup costs, and setup weights. In Section 3.2, we assess the impact of the valid inequalities and the upper bound on the number of bins on the effectiveness of the ILP model. In Section 3.3, we analyze the strength of the LP relaxations, measuring integrality gaps and highlighting structural features of the optimal solutions. The paper concludes with Section 4, which summarizes the contributions and outlines directions for future research.

2. A natural ILP formulation for the BPPS

In this section, we extend the classical *Integer Linear Programming* (ILP) formulation for the BPP (see, e.g., [17]) to the BPPS. To this end, let $k \in \mathbb{Z}_{\geq 1}$ denote an upper bound on the number of bins used in any optimal BPPS solution, and let $\mathcal{B} = \{1, 2, \ldots, k\}$ be the set of candidate bins. The way valid values of k can be obtained is discussed in Section 2.6.

For each item $i \in \mathcal{I}$ and bin $b \in \mathcal{B}$, let x_{ib} be a binary variable equal to 1 if and only if item i is packed into bin b. For each class $c \in \mathcal{C}$ and bin $b \in \mathcal{B}$, let y_{cb} be a binary variable equal to 1 if and only if at least one item of class c is packed into bin b. For each bin $b \in \mathcal{B}$, let z_b be a binary variable equal to 1 if and only if bin b is used. The variables $\boldsymbol{x} \in \{0,1\}^{n\,k}$ define the partition of the items into bins, the variables $\boldsymbol{y} \in \{0,1\}^{m\,k}$ identify the active classes in each bin, and the variables $\boldsymbol{z} \in \{0,1\}^{k}$ indicate which bins are used in an optimal BPPS solution. Using these natural binary variables, we introduce the following ILP formulation for the BPPS, which we refer to as ILP_N, where the subscript "N" stands for natural.

$$(\mathrm{ILP_N}) \qquad \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \quad \sum_{b\in\mathcal{B}} \left(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb} \right) \tag{1a}$$

$$\sum_{b\in\mathcal{B}} x_{ib} = 1, \qquad i \in \mathcal{I}, \tag{1b}$$

$$\sum_{i\in\mathcal{I}} w_i\, x_{ib} + \sum_{c\in\mathcal{C}} s_c\, y_{cb} \le d\, z_b, \qquad b \in \mathcal{B}, \tag{1c}$$

$$x_{ib} \le y_{cb}, \qquad c \in \mathcal{C},\ i \in \mathcal{I}_c,\ b \in \mathcal{B}. \tag{1d}$$

The objective function (1a) to be minimized coincides with the sum of the fixed cost of the used bins and the total setup costs of the classes active in them. The *assignment constraints* (1b) require each item to be packed into exactly one bin. The *capacity constraints* (1c) ensure that the total weight of the items assigned to a bin, plus the setup weights of their active classes, does not exceed the bin capacity; they also guarantee that no items be placed in an unused bin. For each bin $b \in \mathcal{B}$ and item $i \in \mathcal{I}$, the *linking constraints* (1d) enforce that the class of item i is declared active in bin b whenever $x_{ib} = 1$, thereby forcing the corresponding variable y_{cb} to take value 1. Overall, formulation $\mathrm{ILP_N}$ involves $(n + m + 1)\, k$ binary variables and $(n + 1)\, k + n$ linear constraints.

2.1. Strength of the LP relaxations of $\mathrm{ILP_N}$

Let us denote by $\mathrm{LP_N}$ the Linear Programming (LP) relaxation of $\mathrm{ILP_N}$, obtained by replacing the binary variables with the following ones:

$$0 \le x_{ib} \le 1,\ i \in \mathcal{I}, b \in \mathcal{B}, \qquad 0 \le y_{cb} \le 1,\ c \in \mathcal{C}, b \in \mathcal{B}, \qquad 0 \le z_b \le 1,\ b \in \mathcal{B}. \tag{2}$$

We denote by $\zeta(\mathrm{LP_N})$ the optimal solution value of $\mathrm{LP_N}$, which is a valid lower bound on the optimal solution value ψ of the BPPS. The following proposition provides closed-form expressions for both an optimal solution to $\mathrm{LP_N}$ and its optimal value.

Proposition 1. *For every BPPS instance, an optimal solution to* $\mathrm{LP_N}$ *is*

$$x_{ib} = 1/k, \quad i \in \mathcal{I},\ b \in \mathcal{B}, \tag{3a}$$

$$y_{cb} = 1/k, \quad c \in \mathcal{C},\ b \in \mathcal{B}, \tag{3b}$$

$$z_b = \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c}{k\, d}, \quad b \in \mathcal{B}, \tag{3c}$$

and its optimal value is

$$\zeta(\mathrm{LP_N}) = \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c. \tag{4}$$

Proposition 1 shows that an optimal solution to $\mathrm{LP_N}$ is obtained by evenly distributing both the items and the setup weights across the available bins. The fractional value of each z-variable represents the proportion of the bin that is actually utilized. The optimal value of $\mathrm{LP_N}$ is obtained by summing the setup costs of all classes (each counted once) and adding the bin cost multiplied by the ratio between the total item weight plus the total setup weights (each counted once) and the bin capacity.

The following proposition shows that there exist BPPS instances such that, as the number of items n tends to infinity, the *asymptotic performance ratio*, i.e., the ratio between $\zeta(\mathrm{LP_N})$ and ψ, approaches 0. Consequently, the lower bound $\zeta(\mathrm{LP_N})$ can be arbitrarily weak and fails to provide any worst-case performance guarantee. The proof constructs a family of instances in which all items belong to a single class and each bin can contain at most one item due to a very large setup weight. In this case, an optimal BPPS solution requires one bin per item, whereas in the fractional optimal solution of $\mathrm{LP_N}$ both the setup and item weights are fractionally distributed across all bins, yielding a much smaller optimal value.

Proposition 2. *There exists a sequence of BPPS instances such that*

$$\frac{\zeta(\mathrm{LP_N})}{\psi} \longrightarrow 0 \qquad \text{as } n \to \infty.$$

Proof. Consider the family of BPPS instances, defined for each $n \geq 2$, with a single class ($m = 1$, hence $\mathcal{I}_1 = \mathcal{I}$), where

$$w_i = 1, \quad i \in \mathcal{I}, \quad \text{and} \quad s_1 = n - 1.$$

Let $k = n$ and $d = n$, and assume that the setup cost of the class can take any non-negative integer value, i.e., $f_1 \in \mathbb{Z}_{\geq 0}$. Since $w_i + s_1 = n$ for all items $i \in \mathcal{I}$, each bin can contain at most one item. Therefore, any optimal BPPS solution uses one bin per item, and the optimal BPPS value is $\psi = n\,(r + f_1)$. By Proposition 1, the optimal value of $\mathrm{LP_N}$ is

$$\zeta(\mathrm{LP_N}) = \frac{r}{n}\big(n + (n-1)\big) + f_1 = \frac{r}{n}(2n-1) + f_1 = 2r + f_1 - \frac{r}{n}.$$

Taking the limit of the ratio as $n \to \infty$ gives

$$\lim_{n\to\infty} \frac{\zeta(\mathrm{LP_N})}{\psi} = \lim_{n\to\infty} \frac{2r + f_1 - \frac{r}{n}}{n\,(r + f_1)} = \lim_{n\to\infty} \left(\frac{2r + f_1}{n(r + f_1)} - \frac{r}{n^2(r + f_1)}\right) = 0.$$

□

Due to Proposition 1, in the family of instances considered in the proof of Proposition 2, all z-variables take the value

$$\frac{1}{n}\left(\frac{n}{n}+\frac{n-1}{n}\right)=\frac{2n-1}{n^2},$$

representing the fraction of each bin that is actually utilized in an optimal solution to $\mathrm{LP_N}$. Multiplying this value by the bin capacity $d=n$, we obtain that each bin is filled up to the value $(2n-1)/n$. Moreover, in this case the lower bound on the number of used bins, given by the sum of the z-variables, is also $(2n-1)/n$, which converges to 2 as $n \to +\infty$. In contrast, any optimal BPPS solution always requires exactly n bins. An illustration of the difference between BPPS and $\mathrm{LP_N}$ optimal solutions for the family of worst-case instances used in the proof of Proposition 2 is provided in Figure 2.

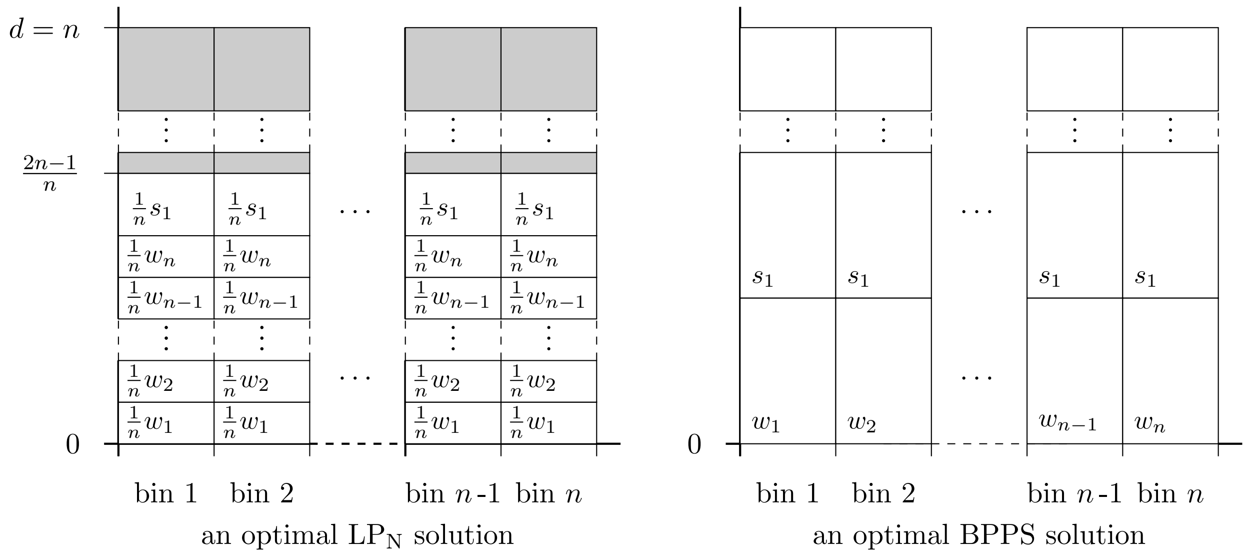

Figure 2: The left part of the figure illustrates an optimal solution of $\mathrm{LP_N}$ for the worst-case instance used in the proof of Proposition 2 with a given value of n. Each bin is filled up to the value $\frac{2n-1}{n}$. The blocks represent the fraction of the item weights and of the class setup weight assigned to each bin. The right part of the figure illustrates an optimal BPPS solution to the same instance, where each bin is completely filled by the weight of one item plus the setup weight of the class.

2.2. Minimum Classes Inequalities

To strengthen $\mathrm{LP_N}$, we introduce the following family of inequalities, called the *Minimum Classes Inequalities* (MCIs):

$$\sum_{b\in\mathcal{B}} y_{cb} \geq \underbrace{\left\lceil \frac{\sum_{i\in\mathcal{I}_c} w_i}{d-s_c} \right\rceil}_{=\gamma_c}, \qquad c\in\mathcal{C}, \tag{5}$$

where, for each class $c \in \mathcal{C}$, the right-hand-side γ_c is a lower bound on the minimum number of bins required to pack all its items. It is defined as the ceiling of the ratio between the total weight of the items in class c and the residual bin capacity $d - s_c$, namely, the capacity available after accounting for the setup weight of class c. These inequalities guarantee that each class is activated in a sufficient number of bins, thereby excluding fractional solutions of LP_N that would otherwise underestimate the actual packing requirements. The right-hand-sides of the MCI (5) can be efficiently computed in linear time with respect to the number of items n of a BPPS instance. The following proposition states that the MCIs in (5) are valid inequalities for ILP_N.

Proposition 3. *For every BPPS instance, the Minimum Classes Inequalities* (5) *are satisfied by all feasible solutions of* ILP_N.

The proof is based on using the constraints of LP_N, together with the integrality of the variables, to derive the MCIs for all integer feasible solutions. We denote by $\text{ILP}_\text{N}^\dagger$ the formulation obtained by adding the MCIs in (5) to ILP_N, and by $\text{LP}_\text{N}^\dagger$ its LP relaxation. We let $\zeta(\text{LP}_\text{N}^\dagger)$ denote the optimal solution value of $\text{LP}_\text{N}^\dagger$, which provides a valid lower bound on the optimal solution value ψ of the BPPS, and is at least as strong as $\zeta(\text{LP}_\text{N})$.

Not only LP_N, but also $\text{LP}_\text{N}^\dagger$ admits closed-form expressions for both an optimal solution to $\text{LP}_\text{N}^\dagger$ and its optimal value, as stated by the following proposition.

Proposition 4. *For every BPPS instance, an optimal solution to* $\text{LP}_\text{N}^\dagger$ *is*

$$x_{ib} = 1/k, \quad i \in \mathcal{I},\ b \in \mathcal{B}, \tag{6a}$$

$$y_{cb} = \gamma_c/k, \quad c \in \mathcal{C},\ b \in \mathcal{B}, \tag{6b}$$

$$z_b = \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{k\, d}, \quad b \in \mathcal{B}, \tag{6c}$$

and its optimal value is

$$\zeta(\text{LP}_\text{N}^\dagger) \ = \ \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) \ + \ \sum_{c\in\mathcal{C}} \gamma_c\, f_c. \tag{7}$$

Proposition 4 shows that an optimal solution of $\text{LP}_\text{N}^\dagger$ is again obtained by evenly distributing the items across the available bins, but now the setup weights of each class $c \in \mathcal{C}$ are distributed proportionally to γ_c. The fractional value of each z-variable represents the proportion of the bin that is actually utilized, and these values are strictly larger than in the solution of Proposition 1 if at least one coefficient γ_c is greater than one. The optimal value of $\mathrm{LP}_\mathrm{N}^\dagger$ is obtained by summing the setup costs of all classes, each counted γ_c times, and adding the bin cost multiplied by the ratio between the total item weight plus the setup weights of all classes, each counted γ_c times, and the bin capacity.

2.3. Strength of the LP relaxations of $\mathrm{ILP}_\mathrm{N}^\dagger$

The inclusion of the MCIs in (5), introduced in Section 2.2, yields the following main theoretical result of the article: the lower bound $\zeta(\mathrm{LP}_\mathrm{N}^\dagger)$ is always strictly larger than 1/2 of the optimal BPPS value ψ. This establishes a worst-case performance guarantee for the lower bound $\zeta(\mathrm{LP}_\mathrm{N}^\dagger)$, proving that $\mathrm{LP}_\mathrm{N}^\dagger$ cannot yield arbitrarily weak lower bounds.

Proposition 5. *For every BPPS instance, we have*

$$\frac{\zeta(\mathrm{LP}_\mathrm{N}^\dagger)}{\psi} > \frac{1}{2}.$$

The proof of Proposition 5 is based on the construction of a heuristic solution to the BPPS, obtained by solving several BPP problems, defined accounting for the setup costs of the classes in different ways: either by reducing the capacity, or by increasing the item weights, or even a mix of the two ideas, depending on the class. This allows the use of classical results on the BPP to show the inequalities sought. The proof is somewhat surprising, since it requires a good control of the structure of the heuristic solution, while simpler constructions with a value which intuitively should be better, do not seem to imply more easily the result.

The following proposition shows that the performance guarantee of 1/2 is tight. In the proof, we provide a family of instances inspired by those proposed in [23, Section 8.3.2]) for the BPP, for which the ratio $\zeta(\mathrm{LP}_\mathrm{N}^\dagger)$ and ψ can be made arbitrarily close to 1/2.

Proposition 6. *There exists a sequence of BPPS instances such that*

$$\frac{\zeta(\mathrm{LP}_\mathrm{N}^\dagger)}{\psi} \longrightarrow \frac{1}{2} \qquad \text{as } d \to \infty.$$

Proof. Consider the family of BPPS instances with a single class ($m = 1$, hence $\mathcal{I}_1 = \mathcal{I}$), defined for each $\vartheta \in \mathbb{Z}_{\geq 1}$ by

$$d = 2\,\vartheta, \qquad w_i = \vartheta, \quad i \in \mathcal{I} \qquad \text{and} \qquad s_1 = 1.$$

We assume that the setup cost of the class can take any non-negative integer value, i.e., $f_1 \in \mathbb{Z}_{\geq 0}$. Since $w_i + s_1 = \vartheta + 1$ for all items $i \in \mathcal{I}$, each bin can contain at most one item. Therefore, any optimal BPPS solution uses one bin per item, and the optimal BPPS value is

$$\psi = n\,(r + f_1).$$

Moreover, we have

$$\gamma_1 \;=\; \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil,$$

and by Proposition 4 the optimal value of $\mathrm{LP}_\mathrm{N}^\dagger$ is

$$\zeta(\mathrm{LP}_\mathrm{N}^\dagger) = \frac{r}{2\,\vartheta}\left(n\,\vartheta + \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil\right) + \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil f_1 = \left(\frac{r}{2\,\vartheta} + f_1\right)\left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil + n\,\frac{r}{2}.$$

Taking the limit of the ratio as $\vartheta \to \infty$ gives

$$\lim_{\vartheta\to\infty} \frac{\zeta(\mathrm{LP}_\mathrm{N}^\dagger)}{\psi} = \lim_{\vartheta\to\infty} \frac{\left(\frac{r}{2\,\vartheta} + f_1\right)\left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil + n\,\frac{r}{2}}{n\,(r + f_1)} = \frac{\frac{n}{2}\,(r + f_1)}{n\,(r + f_1)} = \frac{1}{2}.$$

□

Due to Proposition 4, in the family of instances considered in the proof of Proposition 6, all z-variables take the value

$$\frac{1}{2\,\vartheta}\left(\frac{n\,\vartheta}{n} + \frac{\gamma_1}{n}\right) = \frac{\vartheta + \frac{\gamma_1}{n}}{2\,\vartheta}$$

representing the fraction of each bin that is actually utilized in an optimal solution to $\mathrm{LP}_\mathrm{N}^\dagger$. Multiplying this value by the bin capacity $d = 2\vartheta$, we obtain that each bin is filled up to the value $\vartheta + \frac{\gamma_1}{n}$. Moreover, in this case the lower bound on the number of used bins, given by the sum of the z-variables, is $(n\vartheta + \gamma_1)/(2\vartheta)$, which converges to $n/2$ as $\vartheta \to +\infty$. In contrast, any optimal BPPS solution always requires exactly n bins. An illustration of the difference between BPPS and $\mathrm{LP}_\mathrm{N}^\dagger$ optimal solutions for the worst-case instance used in the proof of Proposition 6 is provided in Figure 3.

Propositions 5 and 6 show that, with the inclusion of the MCIs we can recover the analogous in the BPPS, of the classical results of a tight worst case performance guarantee of $\frac{1}{2}$ for the natural BPP formulation.

2.4. *Minimum Bins Inequality*

To further strengthen $\mathrm{LP}_\mathrm{N}^\dagger$, we introduce the following inequality, called the *Minimum Bins Inequality* (MBI):

$$\sum_{b\in\mathcal{B}} z_b \;\geq\; \underbrace{\left\lceil \frac{\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{d} \right\rceil}_{=\underline{k}}, \tag{8}$$

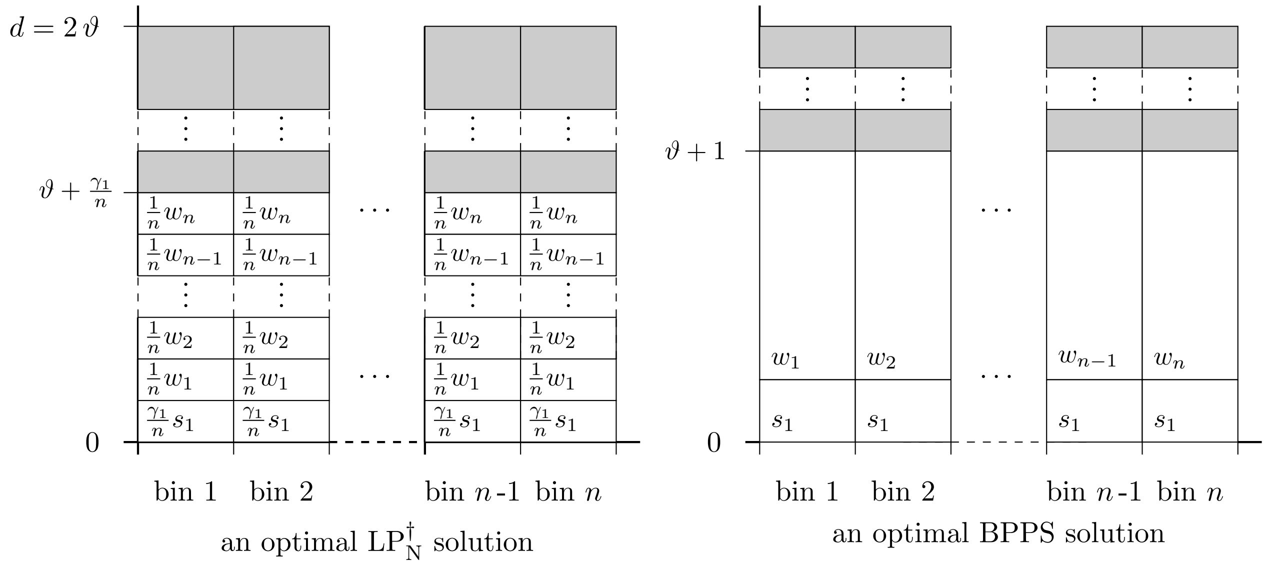
$d = 2\vartheta$
$\vartheta + \frac{\gamma_1}{n}$
$\frac{1}{n}w_n$
$\frac{1}{n}w_{n-1}$
$\frac{1}{n}w_2$
$\frac{1}{n}w_1$
$\frac{\gamma_1}{n}s_1$
0
bin 1 bin 2 bin n-1 bin n
an optimal $\mathrm{LP}^{\dagger}_{\mathrm{N}}$ solution
$\vartheta + 1$
w_1 w_2 w_{n-1} w_n
s_1
0
bin 1 bin 2 bin n-1 bin n
an optimal BPPS solution

Figure 3: The left part of the figure illustrates an optimal solution of $\mathrm{LP}^{\dagger}_{\mathrm{N}}$ for the worst-case instance used in the proof of Proposition 6 with a given value of ϑ. Each bin is filled up to the value $\vartheta + \frac{\gamma_1}{n}$. The blocks represent the fraction of the item weights and of the class setup weight assigned to each bin. The right part of the figure illustrates an optimal BPPS solution to the same instance, where at most one item can be packed in each bin, since two items together with the setup weight do not fit.

where right-hand-side $\underline{k}$ is a lower bound on the minimum number of bins required to pack all items in any optimal BPPS solution while also accounting for setup weights. It is defined as the ceiling of the ratio between the sum of all item weights plus, for each class $c \in \mathcal{C}$, its setup weight s_c counted γ_c times and the bin capacity d. This inequality enforces that the total number of opened bins is sufficient to accommodate both item weights and the class-induced setup weights, thereby excluding fractional solutions of $\mathrm{LP}^{\dagger}_{\mathrm{N}}$ that would otherwise underestimate the overall capacity requirement. The right-hand-side of the MBI (8) can be efficiently computed in linear time with respect to the number of items n of a BPPS instance. The following proposition shows that the MBI in (8) is also a valid inequality for $\mathrm{ILP}_{\mathrm{N}}$.

Proposition 7. *For every BPPS instance, the Minimum Bins Inequality* (8) *is satisfied by every feasible solution of* $\mathrm{ILP}_{\mathrm{N}}$.

The proof is based on using the constraints of LP_{N}, together with the MCIs and the integrality of the z variables, to derive the MBI for all integer feasible solutions.

We denote by $\mathrm{ILP}^{\ddagger}_{\mathrm{N}}$ the formulation obtained by adding the MCIs (5) and the MBI (8) to $\mathrm{ILP}_{\mathrm{N}}$, and by $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ its LP relaxation. We let $\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}})$ denote the optimal solution value of $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$, which provides a valid lower bound on the optimal solution value ψ of the BPPS, and is at least as strong as $\zeta(\mathrm{LP}^{\dagger}_{\mathrm{N}})$.

2.5. Strength of the LP relaxations of $\mathrm{ILP}^{\ddagger}_{\mathrm{N}}$

Not only LP_{N} and $\mathrm{LP}^{\dagger}_{\mathrm{N}}$, but also $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ admits closed-form expressions for both an optimal solution to $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ and its optimal value, as shown by the following proposition.

Proposition 8. *For every BPPS instance, an optimal solution to* $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ *is*

$$x_{ib} = 1/k, \quad i \in \mathcal{I},\ b \in \mathcal{B}, \tag{9a}$$

$$y_{cb} = \gamma_c/k, \quad c \in \mathcal{C},\ b \in \mathcal{B}, \tag{9b}$$

$$z_b = \underline{k}/k, \quad b \in \mathcal{B}, \tag{9c}$$

and its optimal value is

$$\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}}) \;=\; r\,\underline{k} \;+\; \sum_{c\in\mathcal{C}} \gamma_c\, f_c. \tag{10}$$

While the MBI (8) provides a global lower bound on the total number of bins that must be used, the MCIs (5) operate at the class level by enforcing that each class $c \in \mathcal{C}$ is activated in at least γ_c bins. These two families of inequalities are therefore complementary: the MBI (8) constrains only the aggregate number of bins, without distinguishing which classes are responsible for their activation, whereas the MCI (5) ensure that the contribution of each class is explicitly enforced.

Also, the ratio between $\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}})$ and ψ can be made arbitrarily close to 1/2 as shown by the following proposition.

Proposition 9. *There exists a sequence of BPPS instances such that*

$$\frac{\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}})}{\psi} \longrightarrow \frac{1}{2} \qquad \text{as } d \to \infty.$$

The proof employs the same instance of the BPPS which was used in Proposition 6 which shows the analogous result for the model $\mathrm{LP}^{\dagger}_{\mathrm{N}}$.

In our computational experiments of Section 3, we empirically measure the actual improvement of the lower bound provided by $\mathrm{LP}^{\dagger}_{\mathrm{N}}$ and $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ over LP_{N} across the tested instance classes.

2.6. An upper bound to the number of bins used in any optimal BPPS solution

An obvious valid choice for the upper bound k on the number of bins in any optimal BPPS solution is the number of items n, since no solution can use more than one bin per item. This bound, however, is often weak in practice, as the number of bins used is typically much smaller than n. Moreover, the value of k directly determines the number of variables and constraints in the formulation, and thus has a strong impact on computational performance. To address this issue, we develop a stronger, instance-dependent upper bound based on class-wise considerations. Specifically, for each class $c \in \mathcal{C}$, let $\overline{\beta}_c$ denote an upper bound on the minimum number of bins of capacity $d - s_c$ required to pack all its items.

Proposition 10. *For every BPPS instance, the value*

$$\overline{k} = \sum_{c \in \mathcal{C}} \overline{\beta}_c, \tag{11}$$

provides a valid upper bound on the number of bins used in any optimal BPPS solution.

The proof is based on comparing the value of an optimal solution of the BPPS with the value of the first heuristic solutions considered also in the proof of Proposition 5, which is obtained by solving a BPP problem of capacity $d - s_c$ for every class $c \in \mathcal{C}$.

The values $\overline{\beta}_c$, with $c \in \mathcal{C}$, can be determined by applying any algorithm that provides a feasible solution to this BPP associated with class c, since such an algorithm would yield a valid upper bound on the number of bins required. In practice, both exact and heuristic algorithms for the BPP can be employed for this purpose. The specific choices adopted to compute these values in our computational experiments will be discussed in Section 3.

3. Computational experiments

In this section, we present the results of our extensive computational experiments aimed at evaluating the performance of the novel ILP formulation for the BPPS problem introduced in Section 2, along with the three main proposed enhancements: the MCIs (5), described in Section 2.2, the MBI (8), described in Section 2.4, and the upper bound on the number of bins that can be used in any optimal BPPS solution, described in Section 2.6.

We are interested in determining optimal BPPS solutions by solving the ILP model with a state-of-the-art solver. When optimality cannot be certified within the imposed time limit, we evaluate the solution quality through the optimality gap. Our experiments pursue three main goals: first, to assess the computational effectiveness of the proposed ILP formulation and its variants by identifying the largest instance sizes (and their structural features) that can be solved to optimality within a given time limit, determining the features that most significantly affect problem difficulty, and evaluating the relative performance of different formulation enhancements (see Section 3.2); second, to empirically evaluate the strength of the LP relaxation bounds with and without the inclusion of MCIs (5) and MBI (8), thereby complementing the theoretical results discussed in Sections 2.1 and 2.3 (see Section 3.3); and third, to analyze the structure of the optimal BPPS solutions in terms of the number of items and classes per used bin, as well as bin utilization levels, in order to highlight their main characteristics (also in Section 3.3).

To the best of our knowledge, the BPPS is a novel problem that has not been previously studied in the literature. Consequently, no existing models or algorithms are available for comparison with our proposed ILP formulation. Furthermore, no publicly available benchmark library exists. For this reason, we designed a diverse and systematic set of test instances that capture the main structural features of BPPS instances, which we describe next.

3.1. Library of benchmark BPPS instances

In this section, we introduce the library of benchmark instances specifically designed for the BPPS. This testbed aims to provide a diverse and representative collection of instances, varying in size and structural characteristics, enabling a comprehensive evaluation of the proposed ILP formulation and an assessment of how different instance features influence computational performance.

To generate the instances, we extended the approach proposed in Schwerin and Wäscher [32] for the classical BPP, thereby adopting a methodology consistent with the literature on bin packing problems, while incorporating item classes along with their associated setup weights and costs. Each BPPS instance $I(BPPS)$ is defined by a tuple $(n, \boldsymbol{w}, d, m, \mathcal{P}, \boldsymbol{s}, \boldsymbol{f}, r)$. While most of these input data can be freely chosen, item weights and class setup weights must be generated so as to guarantee feasibility. In line with [32], we introduce parameters $\upsilon_1, \upsilon_2, \sigma_1, \sigma_2 \in [0,1]$, with $\upsilon_1 < \upsilon_2$ and $\sigma_1 < \sigma_2$, which determine the sampling ranges for item and setup weights, respectively. Specifically, for each $i \in \mathcal{I}$, the item weight is sampled uniformly from the integer values in the set $[\upsilon_1\, d,\ \upsilon_2\, d] \cap \mathbb{Z}$, and for each $c \in \mathcal{C}$, its setup weight is sampled uniformly in the set $[\sigma_1\, d,\ \sigma_2\, d] \cap \mathbb{Z}$.

To ensure instance feasibility, we enforce the condition $\upsilon_2 + \sigma_2 \leq 1$. Each item is assigned to a class, selected uniformly at random among the m available classes. Every instance is is created either with or without setup and bin costs. In the former case, setup costs are sampled uniformly in the set $[1, r/2] \cap \mathbb{Z}$ for each $c \in \mathcal{C}$, with the bin cost set to $r = 10$. In the latter, setup costs are set to zero, and the bin cost is set to $r = 1$. This latter configuration corresponds to minimizing the number of used bins, since no setup costs are present.

We generated BPPS instances for all combinations of the following instance-generator parameters: the number of items $n \in \{25, 50, 75, 100, 200\}$, the number of classes $m \in \{5, 10\}$, the bin capacity $d \in \{200, 1000, 10000\}$, the cost structure (with or without setup costs and bin costs), two item size categories—*small* ($\upsilon_1 = 0.05$, $\upsilon_2 = 0.15$) and *large* ($\upsilon_1 = 0.15$, $\upsilon_2 = 0.30$)—and two setup weight categories—*small* ($\sigma_1 = 0.01$, $\sigma_2 = 0.10$) and *large* ($\sigma_1 = 0.10$, $\sigma_2 = 0.20$). For each combination of these parameters, two instances were generated using different random seeds, resulting in a total of 480 instances. The complete benchmark set and its documentation, along with the generator toolkit, are publicly available at the following GitHub repository: `https://github.com/FabioCiccarelli/BPPS_instances.git`. We hope that this challenging library of instances will foster further research on the BPPS and support the development and comparison of novel exact and heuristic algorithms.

3.2. Computational performance of the ILP formulation

In this section, we compare the computational performance of the following four variants of the proposed ILP formulation for the BPPS:

1. The baseline, denoted by ILP_N, is the compact formulation in which the upper bound k on the number of bins in any optimal BPPS solution is set to n, the number of items.

2. The first variant, denoted by $\text{ILP}^{\dagger}_\text{N}$, extends ILP_N by including the MCIs (5).

3. The second variant, denoted by $\text{ILP}^{\ddagger}_\text{N}$, extends ILP_N by including the MCIs (5) and the MBI (8).

4. The third variant, denoted by $\text{ILP}^{\star}_\text{N}$, extends $\text{ILP}^{\ddagger}_\text{N}$ by including the upper bound $\overline{k}$ on the number of bins in any optimal BPPS solution (see Proposition 10). The values $\overline{\beta}_c$, for each class $c \in \mathcal{C}$, used to compute $\overline{k}$ were obtained through classical heuristic algorithms for the BPP. Specifically, we applied the Next Fit (NF), First Fit (FF), and Best Fit (BF) algorithms (see [21]), each executed under 50 random permutations of the item order (including the non-increasing order of weights).

All tests were conducted on an Ubuntu machine equipped with an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz and 256 GB of RAM. The ILP formulations are implemented in `C++` and solved using *IBM ILOG CPLEX v22.1.1*, running in single-thread mode with default settings. A time limit of 1800 seconds is imposed for each test.

Table 1 reports, for each of the four ILP formulation variants presented above, the number of instances solved to proven optimality within a time limit of 1800 seconds (columns "#opt") and the average optimality gap (columns "gap") over the instances not solved to optimality. The optimality gap is defined as the percentage difference between the best upper bound and the best lower bound computed by the solver during the solution process for a given formulation, taken with respect to the best upper bound.

The instances are grouped according to the six instance-generator parameters described in the previous section on the benchmark instance library and, for each group, the table also reports the number of instances in that group (column "#inst").

The results in Table 1 show that the performance differences between the four variants become more pronounced as the instance size or complexity increases. Indeed, for small instances (e.g., $n = 25$), all four formulations perform similarly. However, as n grows, the baseline ILP_N rapidly loses effectiveness, with a sharp drop in the number of optimally solved instances and a substantial increase in the gap. The addition of the MCI constraints in $\text{ILP}_\text{N}^\dagger$ yields a clear improvement; similarly, the inclusion of the MBI enhances the model performance and allows to solve to full optimality a significantly higher number of instances. Finally, the combination of these two improvements and the upper bound on the number of bins in $\text{ILP}_\text{N}^\star$ consistently provides the best results, both in terms of the number of optimal solutions and reduced gap, with the only exception of the set of instances with $n = 200$, where $\text{ILP}_\text{N}^\ddagger$ allows to optimally solve one more instance than $\text{ILP}_\text{N}^\star$. A similar trend is observed when increasing the number of classes m, where $\text{ILP}_\text{N}^\star$ remains the strongest variant, followed by $\text{ILP}_\text{N}^\ddagger$. The bin capacity d does not seem to have a significant impact on the performance of the four variants. Nevertheless, in line with previous findings, $\text{ILP}_\text{N}^\star$ maintains a clear advantage over the others across all tested values. The same pattern emerges for the remaining generation parameters: with and without setup costs, for both small and large setup weights, and for small and large item weights, $\text{ILP}_\text{N}^\star$ systematically outperforms the other three variants. In particular, the improvements of the enhanced variants over ILP_N are substantial for the most challenging settings (large n or large m), indicating that the proposed strengthening techniques significantly increase

Table 1: Performance comparison of the four variants of the ILP formulation for the BPPS, reporting the number of instances solved to optimality and the average optimality gap. The instances are grouped by instance generator parameters. Time limit: 1800 seconds.

			ILP_N		$\text{ILP}^\dagger_\text{N}$		$\text{ILP}^\ddagger_\text{N}$		$\text{ILP}^\star_\text{N}$	
param		#inst	#opt	gap	#opt	gap	#opt	gap	#opt	gap
`item number`	$n = 25$	96	92	5.8	94	6.6	95	6.7	96	-
	$n = 50$	96	36	9.5	56	7.0	63	6.4	68	5.8
	$n = 75$	96	25	10.8	35	6.6	41	5.6	47	5.7
	$n = 100$	96	9	14.0	25	6.4	35	5.7	39	6.0
	$n = 200$	96	0	20.9	10	6.7	19	6.4	18	5.5
`class number`	$m = 5$	240	89	14.1	128	6.0	144	5.5	154	4.9
	$m = 10$	240	73	14.7	92	7.1	109	6.4	114	6.2
`bin capacity`	$d = 200$	160	58	14.5	76	6.8	81	5.9	88	5.4
	$d = 1000$	160	50	14.2	69	6.6	89	5.9	89	5.8
	$d = 10000$	160	54	14.5	75	6.6	83	6.2	91	5.9
`bin-setup costs`	no	240	97	12.0	124	8.0	120	6.8	125	6.5
	yes	240	65	16.4	96	5.6	133	5.1	143	4.8
`item weights`	large	240	50	15.4	65	6.8	74	6.1	83	5.7
	small	240	112	13.0	155	6.4	179	5.8	185	5.6
`setup weights`	large	240	71	15.7	99	7.2	109	6.5	122	6.0
	small	240	91	13.0	121	6.1	144	5.4	146	5.3
Total/Average		480	162	14.4	220	6.7	253	6.0	268	5.7

scalability and solution quality. Overall, across all 480 benchmark instances, ILP_N, $\text{ILP}^\dagger_\text{N}$, $\text{ILP}^\ddagger_\text{N}$ and $\text{ILP}^\star_\text{N}$ solve 162, 220, 253 and 268 instances to optimality, respectively, with an average optimality gap on the unsolved instances of 14.4%, 6.7%, 6.0% and 5.7%, further confirming the superiority of $\text{ILP}^\star_\text{N}$.

Figure 4 shows the distribution of computing times (in seconds) for the four ILP variants, evaluated on the subset of instances solved to proven optimality by $\text{ILP}^\star_\text{N}$. Each box plot reports the median, the interquartile range, and the extreme values, together with markers for the mean, thus providing insights into both central tendency and variability. The results highlight that $\text{ILP}^\star_\text{N}$ consistently achieves the lowest mean times across all tested values of n, also having the lowest median times in most of the cases. For example, for instances with $n = 25$, the median time for $\text{ILP}^\star_\text{N}$ is around 0.16 seconds, which is approximately 7 times lower than that required by ILP_N. This spread significantly widens as the item number grows: in the instances with $n = 75$ solved to optimality by ILP_N, $\text{ILP}^\star_\text{N}$ has a median solution time of just under 10 seconds, slightly above that of $\text{ILP}^\ddagger_\text{N}$. In contrast, $\text{ILP}^\dagger_\text{N}$ has a median time slightly above 30 seconds, and ILP_N has a median time of more than 930 seconds. For benchmark instances with $n = 200$, with the baseline formulation ILP_N we fail to solve any instance to optimality. In contrast, $\text{ILP}^\dagger_\text{N}$ requires a mean solution time exceeding 1300 seconds, $\text{ILP}^\ddagger_\text{N}$ on average requires more than 750 seconds and $\text{ILP}^\star_\text{N}$ maintains its superior performance, with a mean time of under 300 seconds. Moreover, for this last group of instances, the first quartile of $\text{ILP}^\star_\text{N}$'s solution time's distribution is just around 17 seconds, indicating that 25% of these large instances can still be efficiently solved.

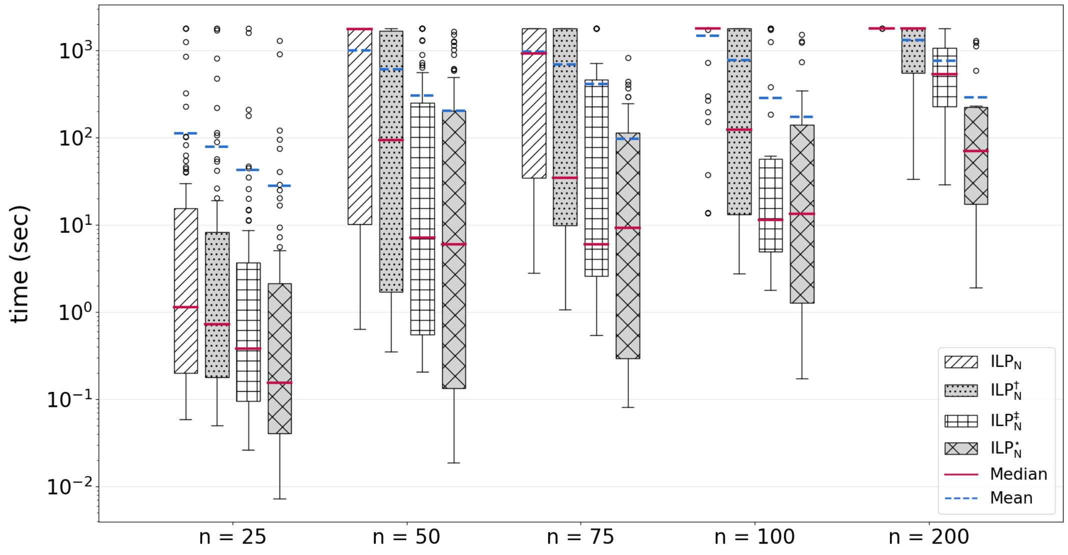

Figure 4: Box plot of the solution times of the different variants of the ILP formulation for the BPPS, for the instances solved to optimality by $\text{ILP}^\star_\text{N}$. The vertical axis is in logarithmic scale. Time limit: 1800 seconds.

Figure 5 shows the distribution of the optimality gap (in %) for the four ILP formulation variants, computed over all instances not solved to proven optimality within the time limit by $\text{ILP}^\star_\text{N}$. For each variant, the corresponding box plot shows the median, the interquartile range, and the extreme values, thus illustrating both the gap's central tendency and its variability across instances. From the figure, it emerges that the enhanced formulations $\text{ILP}^\dagger_\text{N}$, $\text{ILP}^\ddagger_\text{N}$ and $\text{ILP}^\star_\text{N}$ achieve significantly smaller gaps on average, with substantially lower dispersion compared to the baseline ILP_N. For example, for instances with $n = 100$, ILP_N shows a median gap slightly below 15%, while the other variants are all below half of such value. For the largest instances ($n = 200$), $\text{ILP}^\star_\text{N}$ maintains a median gap close to 5%, with a very narrow interquartile range from approximately 4% to 7%.

Conversely, the median gap of $\mathrm{ILP_N}$ increases to more than 20%, and its dispersion is considerable, with an interquartile range ranging from approximately 15% to more than 28%.

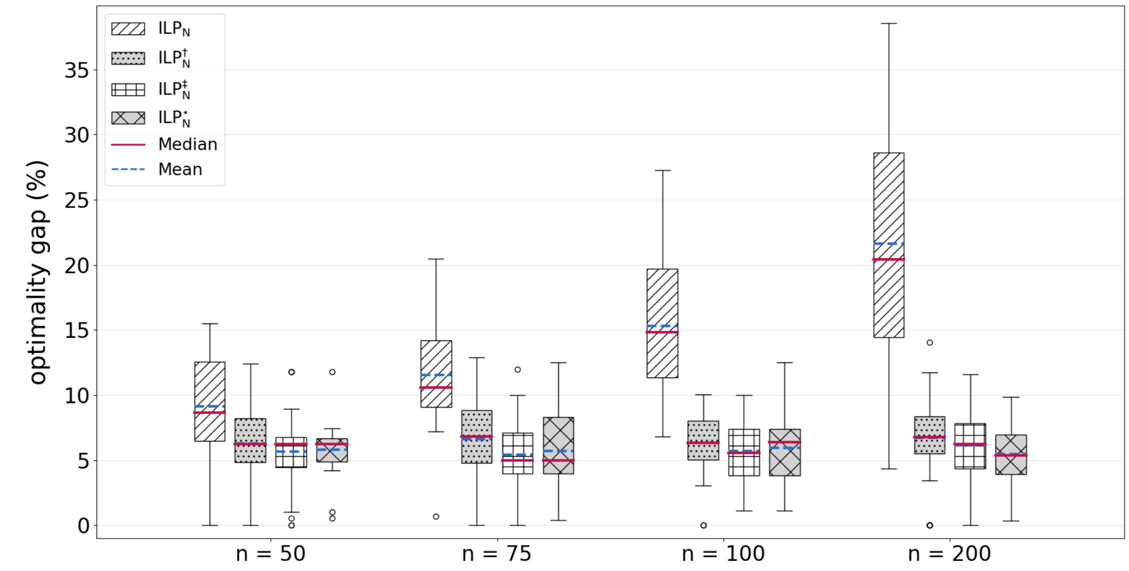

Figure 5: Box plot of the optimality gap of the different variants of the ILP formulation for the BPPS for the instances not solved to optimality within the time limit by $\mathrm{ILP_N^\star}$. Time limit of 1800 seconds.

Figure 6 plots the number of instances solved as a function of time for the four ILP formulation variants. The x-axis represents the computing time (in seconds) and the y-axis reports the number of instances solved to optimality within that time. Each curve, therefore, shows the cumulative solving profile of a variant, enabling a direct visual comparison of their efficiency.

The plot clearly indicates that the best-performing variant $\mathrm{ILP_N^\star}$ consistently dominates the others, solving a larger fraction of instances across all time thresholds. More specifically, within the first 0.1 seconds, $\mathrm{ILP_N^\star}$ outperforms the other three by solving almost 60 instances, whereas $\mathrm{ILP_N}$, $\mathrm{ILP_N^\dagger}$ and $\mathrm{ILP_N^\ddagger}$ solve only 5, 10 and 25 instances, respectively. This performance gap persists by the 1-second mark, at which point $\mathrm{ILP_N^\star}$ has solved roughly 120 instances, compared to 85 for $\mathrm{ILP_N^\ddagger}$, 66 for $\mathrm{ILP_N^\dagger}$ and 54 for $\mathrm{ILP_N}$. For harder instances, the gap significantly decreases: at the 100-second threshold, $\mathrm{ILP_N^\star}$ has increased its count to 210 solved instances, with $\mathrm{ILP_N^\ddagger}$ at 206, $\mathrm{ILP_N^\dagger}$ at 172 and $\mathrm{ILP_N}$ at 131. Beyond the cumulative counts, the slope of each profile reveals the rate at which each variant solves instances over time. The initial steep ascent of the curve for $\mathrm{ILP_N^\star}$ confirms its remarkable ability to quickly solve a large number of easier instances. In the interval between 0.1 and 1 second, it solves an additional 60 instances, demonstrating the highest solve rate among the four variants in this early phase, together with $\mathrm{ILP_N^\ddagger}$. In the subsequent phase (from 1 to 10 seconds), the solve

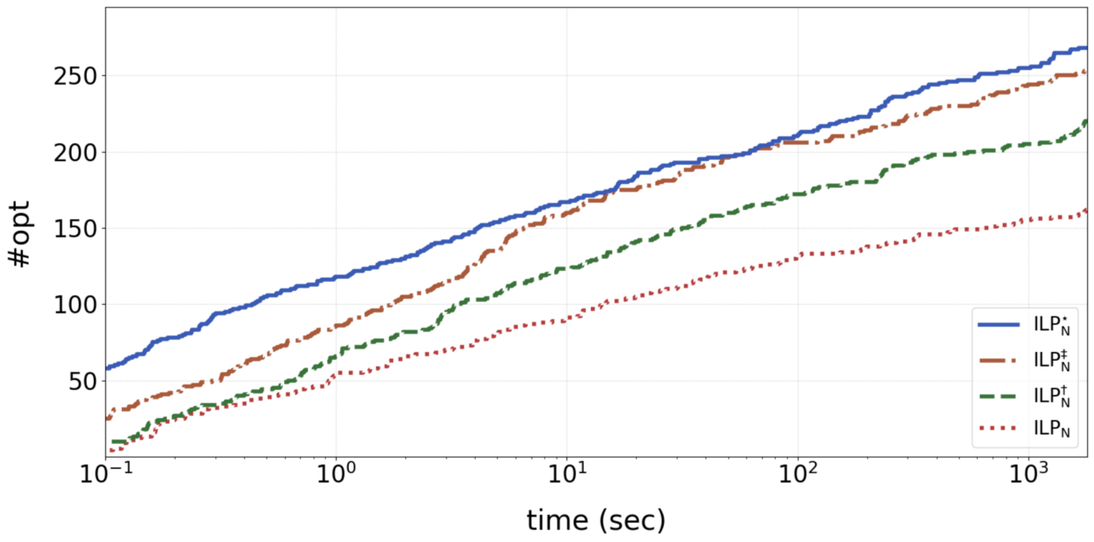

Figure 6: Number of instances solved as a function of time for the different variants of the ILP formulation for the BPPS. Time limit: 1800 seconds. The horizontal axis is in logarithmic scale.

rate of $ILP^{\star}_{N}$ moderates to 49 new instances, while $ILP^{\ddagger}_{N}$ reaches its peak efficiency, solving 75 new instances in the same interval. Finally, for harder instances requiring longer solution times (from 100 to 1000 seconds), all variants exhibit a decelerating solve rate. However, $ILP^{\star}_{N}$ and $ILP^{\ddagger}_{N}$ maintain a clear edge, solving 45 and 38 new instances, respectively, while the ILP_{N} and $ILP^{\dagger}_{N}$ struggle, solving only 24 and 32 additional instances, respectively. This confirms that the enhanced formulations, particularly $ILP^{\star}_{N}$, are not only faster on average but are also more capable of making progress on difficult instances within the given time constraints.

3.3. Strength of the LP relaxations and features of the optimal solution

In this section, we compare the root-node bound obtained with the variants of the formulation we proposed and analyze the main features of the solutions that we obtained on our test set. The results are reported in Table 2. The table is organized into two main parts and reports aggregate statistics for the subset of instances for which at least one of the tested formulations found a certified optimal solution. This subset is chosen because it provides a reliable ground truth against which the quality of bounds and the solution features can be accurately measured. The analysis includes a total of 286 instances: while $ILP^{\star}_{N}$ demonstrates superior overall performance by solving 268 instances to optimality, the remaining formulations contribute additional solutions for instances where $ILP^{\star}_{N}$ fails to converge within the time limit of 1800 seconds. Specifically, among the instances not solved optimally by $ILP^{\star}_{N}$, formulation $ILP^{\ddagger}_{N}$ successfully solves 9 instances, $ILP^{\dagger}_{N}$ solves 13 instances, and even the baseline formulation $\mathrm{ILP_N}$ manages to solve one instance that remains unsolved by $\mathrm{ILP_N^\star}$. The union of instances solved optimally by $\mathrm{ILP_N^\star}$ and those solved by at least one of the remaining variants constitutes the complete set of 286 instances with certified optimal solutions. For each subset of the optimally solved instances, grouped by different instance-generator parameter values, the first part of Table 2 (columns *Integrality gaps*) reports the integrality gap, defined as the percentage difference between the optimal BPPS value and the optimal value of the LP relaxation for a given formulation, taken with respect to the optimal BPPS value. We consider the baseline formulation $\mathrm{ILP_N}$, whose optimal LP relaxation solution value is $\zeta(\mathrm{LP_N})$ and the enhanced formulations $\mathrm{ILP_N^\dagger}$, $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$, whose optimal LP relaxation solution values are $\zeta(\mathrm{LP_N^\dagger})$, $\zeta(\mathrm{LP_N^\ddagger})$ and $\zeta(\mathrm{LP_N^\star})$, respectively. As established in Equation (10), the upper bound $\overline{k}$ does not affect the optimal solution value of the LP relaxation for $\mathrm{ILP_N^\star}$. Consequently, the optimal values for the relaxations of $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$ are identical, i.e., $\zeta(\mathrm{LP_N^\star}) = \zeta(\mathrm{LP_N^\ddagger})$. Therefore, the remainder of this section will focus exclusively on the value of $\zeta(\mathrm{LP_N^\star})$.

The first part of the table (*Integrality gap*) allows us to directly quantify the strengthening effect of the MCIs and that of the MBI across different instance-generator parameters. The results demonstrate that incorporating the MCIs significantly reduces the integrality gap across all problem settings compared to formulation $\mathrm{ILP_N}$. This improvement becomes even more pronounced when the MBI is included in formulations $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$. For larger instances with $n = 200$, the average integrality gap decreases from 19.5% for formulation $\mathrm{ILP_N}$ to 2.7% for $\mathrm{ILP_N^\dagger}$, and further reduces to zero for $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$. Furthermore, When increasing the number of items n, the gap of $\mathrm{ILP_N}$ steadily grows while that of the other three variants significantly decreases. Conversely, when increasing the number of classes from $m = 5$ to $m = 10$, the average integrality gap of $\mathrm{ILP_N}$ decreases, while those of the other variants increase. Nevertheless, $\mathrm{ILP_N^\dagger}$ and $\mathrm{ILP_N^\star}$ still maintain a clear advantage over $\mathrm{ILP_N}$. Changes in bin capacity d have only a marginal effect on the integrality gaps, suggesting that this parameter is less critical for bound quality. The presence of setup costs significantly increases the gap for $\mathrm{ILP_N}$ (up to 19.9%), whereas $\mathrm{ILP_N^\dagger}$ and $\mathrm{ILP_N^\star}$ reduce it to 6.7% and 2.5%, respectively. This can be attributed to the fundamental difference in how the formulations handle class setup costs within their LP relaxation optimal value. Indeed, the optimal solution value of $\mathrm{LP_N}$ accounts for each class exactly once in the objective function (see Equation (4)), regardless of how many bins actually contain items of that class. In contrast, formulations $\mathrm{ILP_N^\dagger}$, $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$, incorporating the MCIs, include an estimate of the number of times each class $c \in \mathcal{C}$ is activated across different bins, namely the γ_c coefficient (see Equations (7) and (10), respectively). Consequently, when setup

Table 2: Quality of the lower bound provided by the LP relaxation of the variants of the ILP formulation for the BPPS, and features of optimal BPPS solutions. The reported values refer to the subset of instances for which at least one formulation found a certified optimal solution.

			Integrality gap					Optimal BPPS solution features			
param		#inst	ILP$_\text{N}$	ILP$^\dagger_\text{N}$	ILP$^\star_\text{N}$	$\underline{k}$	$\overline{k}$	#bins	#items	#classes	%fill
`item number`	$n = 25$	96	16.0	12.8	4.2	5.5	8.7	5.8	4.9	1.8	87.2
	$n = 50$	72	16.6	7.9	3.0	9.1	12.0	9.4	6.0	1.5	92.6
	$n = 75$	52	16.4	4.5	0.2	12.1	14.9	12.1	6.9	1.4	95.5
	$n = 100$	41	17.6	4.5	0.2	13.9	16.3	14.0	7.6	1.3	95.6
	$n = 200$	25	19.5	2.7	0.0	22.9	25.2	22.9	8.7	1.2	97.2
`class number`	$m = 5$	164	19.0	7.1	1.7	11.2	13.1	11.3	6.3	1.3	92.7
	$m = 10$	122	13.7	9.1	3.0	9.1	13.3	9.4	6.1	1.8	91.4
`bin capacity`	$d = 200$	93	16.1	7.7	2.1	10.0	12.8	10.2	6.5	1.5	92.5
	$d = 1000$	96	17.0	8.2	2.5	10.4	13.2	10.6	6.2	1.5	92.0
	$d = 10000$	97	17.2	8.0	2.1	10.6	13.5	10.8	6.1	1.5	91.9
`bin-setup costs`	no	138	13.5	9.3	1.9	10.0	12.8	10.1	6.2	1.6	92.8
	yes	148	19.9	6.7	2.5	10.7	13.6	10.9	6.3	1.4	91.5
`item weights`	large	90	17.8	7.3	2.4	11.8	14.4	12.1	3.5	1.4	93.2
	small	196	16.3	8.3	2.1	9.7	12.6	9.8	7.5	1.6	91.7
`setup weights`	large	129	19.5	9.0	3.2	10.3	13.0	10.6	5.7	1.4	91.4
	small	157	14.5	7.1	1.4	10.3	13.4	10.4	6.7	1.6	92.8
Total/Average		286	16.8	8.0	2.2	10.3	13.2	10.5	6.3	1.5	92.1

costs are present, the LP relaxation of ILP$_\text{N}$ fails to capture this term of the objective function, as it underestimates the total setup cost by not accounting for multiple activations of the same class. As far as the remaining instance-generator parameters are concerned, item weights do not seem to significantly affect the integrality gap values, while small setup weights tend to reduce the gap for all the variants considered.

The second part of the table includes the average values of the lower bound $\underline{k}$ (see Proposition 8) and the upper bound $\overline{k}$, computed as detailed at the beginning of Section 3.2, as well as the structural characteristics of the optimal solutions found (*Optimal BPPS Solution Features*): specifically, these last columns report the average number of bins used in the optimal solution (#bins), and other features such as items per bin, active classes per bin, and bin fill percentage. The effectiveness of the bounds $\underline{k}$ and $\overline{k}$ can be assessed by comparing their average values to the actual average number of bins required in optimal BPPS solutions. Notably, the upper bound $\overline{k}$ provides a loose estimate of the optimal bin count across all problem categories. For example, for instances with $m = 10$, its average value is 13.3 while the average number of actually used bins is 9.4, yielding a percentage difference exceeding 40%. Despite this imprecision, $\overline{k}$ remains substantially smaller than the trivial bound n across all instance sizes, with the most significant reduction observed for larger instances with $n = 200$. This reduction enables a substantial decrease in the number of variables and constraints within formulation $\text{ILP}_{\text{N}}^{\star}$, thereby explaining its superior computational performance relative to the other variants. In contrast, the lower bound $\underline{k}$ demonstrates high accuracy, with values closely matching the actual number of bins used in optimal BPPS solutions. Across all instance-generator parameter values, the difference between $\underline{k}$ and the actual number of bins used does not exceed 0.3, with this difference approaching zero for larger instances with $n = 200$. This precision provides additional evidence for the computational performance improvements observed when incorporating the MBI into formulations $\text{ILP}_{\text{N}}^{\ddagger}$ and $\text{ILP}_{\text{N}}^{\star}$. The solution statistics also reveal that optimal solutions typically contain between 6 and 7 items per bin, involve a small number of active classes (around 1.5), and achieve a high utilization rate of the bin capacity (92% on average) across all parameter configurations. When the number of items n increases, the average number of items per bin grows significantly, while the number of active classes per bin decreases, reflecting a tendency towards more homogeneous bin composition in larger instances. Conversely, increasing the number of classes m results in slightly more diverse bins (higher number of classes per bin), whereas variations in bin capacity d have minimal impact on these statistics. The presence of setup costs, while not affecting significantly the number of items per bin, is associated with marginally fewer classes per bin, suggesting a packing strategy that limits costly setups. Small item weights produce solutions with more items per bin and a higher diversity of classes, while large setup weights lead to both fewer items per bin and slightly lower diversity of classes. Overall, these patterns indicate that optimal solutions exploit bin capacity efficiently while adapting class composition to the cost and weight structure of the instance.

To investigate whether state-of-the-art solvers can systematically improve upon LP-based lower bounds through their internal cutting plane procedures, we conducted an additional experiment by solving the baseline formulation ILP_{N} at the root node only, without branching, and comparing the lower bound obtained by CPLEX after its internal cut generation with the optimal value of the LP relaxation LP_{N}, which can be computed in closed form as in Equation (4). Out of 480 instances, in 465 cases (96.9%) the two values coincide perfectly, indicating that CPLEX's cut generation routines provide no improvement over the baseline LP bound. In the remaining 15 instances (3.1%), the solver identifies the optimal solution at the root node because the incumbent value matches the ceiling of the LP relaxation value. These results demonstrate that CPLEX's general-purpose cutting plane algorithms fail to achieve the substantial lower bound improvements provided instead by the MCIs and the MBI, and underscore the importance of the problem-specific valid inequalities we propose in this paper, which are essential for achieving high-quality formulations.

Finally, we conducted an analysis comparing the LP relaxation bound $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$ obtained using the right-hand-side γ_c for the MCIs, as in Equation (5), against that obtained using the optimal BPP solution values for each class $c \in \mathcal{C}$ (denoted by β_c). The values $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ differ in only 40 instances out of 480, demonstrating that the $\boldsymbol{\gamma}$ coefficients provide highly accurate estimates of the minimum number of bins required to pack all items of each class $c \in \mathcal{C}$. In the instances where these values differ, this results in increased right-hand-side values for the MCIs (5) and a consequent improvement in the lower bound $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$. However, the impact remains minimal: across these 40 cases, the average percentage difference between the two lower bounds is around 1.1%. Also when considering the subset of 286 optimally solved instances reported in Table 2, the absolute difference between the average LP bound $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$ obtained with the $\boldsymbol{\gamma}$ values and that obtained with the optimal $\boldsymbol{\beta}$ values is only 0.05 percentage points. These results underscore the ability of the lower bounds $\boldsymbol{\gamma}$ to approximate the optimal BPP solution values for individual classes, and their usefulness for obtaining a tight integrality gap without the computational overhead of solving m separate BPP instances. Similarly, regarding the upper bound $\overline{k}$ (11), we compared the value $\overline{k}$ obtained summing the heuristic class-wise solution values $\overline{\beta}_c$, computed as detailed at the beginning of Section 3.2, with that obtained summing the optimal BPP solution values β_c for each class $c \in \mathcal{C}$. These two upper bounds coincide in 434 out of 480 instances. In the remaining 46 instances, concentrated primarily among larger instances with $n = 200$ (which account for 30 of the 46 cases), the average percentage difference between the two upper bounds is around 3.5%. This confirms that the heuristic estimation of the $\overline{\boldsymbol{\beta}}$ values for computing the upper bound $\overline{k}$ is highly accurate, and even when it produces a looser upper bound $\overline{k}$ compared to the one obtained using the optimal $\boldsymbol{\beta}$ values, the difference remains marginal and does not lead to a significant reduction in the number of variables and constraints of formulation $\mathrm{ILP}_{\mathrm{N}}^{\star}$.

4. Conclusions

In this paper, we introduced and studied the *Bin Packing Problem with Setups* (BPPS), a novel generalization of the classical Bin Packing Problem in which items are partitioned into classes and, whenever an item from a given class is packed into a bin, a setup weight and cost are incurred. This setting models a wide range of practical applications where setup operations are required, particularly in production planning and logistics. We proposed a natural ILP formulation for the BPPS and analyzed the structural properties of its LP relaxation, which admits a closed-form optimal solution but may yield arbitrarily weak lower bounds. To strengthen the relaxation, we introduced a new family of valid inequalities, the *Minimum Classes Inequalities* (MCIs), and proved that their inclusion guarantees a worst-case performance ratio of 1/2. We also introduced the *Minimum Bins Inequalities* (MBI), which further improved the quality of the LP relaxation. To support computational evaluation, we developed a publicly available benchmark library of 480 BPPS instances, designed to capture a broad spectrum of instance characteristics, including variations in item weights, class distributions, setup costs, and setup weights. Using this library, we carried out an extensive computational campaign to assess the performance of the enhancements proposed for the ILP formulation. Our experiments show that the enhanced model with MCIs and MBI, combined with a tight upper bound on the number of bins, substantially improves overall computational performance. The strengthened formulation consistently solves a larger number of instances to proven optimality compared to the baseline. Moreover, for instances that remain unsolved within the time limit, it delivers very small optimality gaps, confirming its practical effectiveness even on large and challenging instances.

Future research directions include investigating the polyhedral structure of the proposed formulation, developing a branch-and-price algorithm to solve a set-covering formulation of the problem via column generation, and designing tailored exact and heuristic algorithms that exploit the specific role of class-induced setup constraints. Another promising avenue is to study the approximability of the problem and to develop polynomial-time approximation algorithms with provable performance guarantees.

References

[1] U. Akinc. Approximate and exact algorithms for the fixed-charge knapsack problem. *European Journal of Operational Research*, 170(2):363–375, 2006.

[2] N. Altay, P. R. Jr., and K. Bretthauer. Exact and heuristic solution approaches for the mixed integer setup knapsack problem. *European Journal of Operational Research*, 190(3):598–609, 2008.

[3] R. Baldacci, S. Coniglio, J.-F. Cordeau, and F. Furini. A numerically exact algorithm for the bin-packing problem. *INFORMS Journal on Computing*, 36(1):141–162, 2024.

[4] M. Barkel, M. Delorme, E. Malaguti, and M. Monaci. Bounds and heuristic algorithms for the bin packing problem with minimum color fragmentation. *European Journal of Operational Research*, 320(1):57–68, 2025.

[5] D. Bergman, C. Cardonha, and S. Mehrani. Binary decision diagrams for bin packing with minimum color fragmentation. In L. Rousseau and K. Stergiou, editors, *CPAIOR 2019*, volume 11494 of *LNCS*, pages 57–66. Springer, 2019.

[6] R. Bishara. Cold chain management - An essential component of the global pharmaceutical supply chain. *American Pharmaceutical Review*, 9:105–109, 01 2006.

[7] Y. G. Borges, F. K. Miyazawa, R. C. Schouery, and E. C. Xavier. Exact algorithms for class-constrained packing problems. *Computers & Industrial Engineering*, 144:106455, 2020.

[8] P. E. Brecht, J. K. Brecht, and J. E. Saenz. Chapter 18 - Temperature-controlled transport for air, land, and sea. In E. M. Yahia, editor, *Postharvest Technology of Perishable Horticultural Commodities*, pages 591–637. Woodhead Publishing, 2019.

[9] M. Casazza and A. Ceselli. Mathematical programming algorithms for bin packing problems with item fragmentation. *Computers & Operations Research*, 46:1–11, 2014.

[10] A. Ceselli and G. Righini. An optimization algorithm for the ordered open-end bin-packing problem. *Operations Research*, 56(2):425–436, 2008.

[11] E. Chajakis and M. Guignard. Exact algorithms for the setup knapsack problem. *INFOR: Information Systems and Operational Research*, 32(3):124–142, 1994.

[12] K. Chebil and M. Khemakhem. A dynamic programming algorithm for the knapsack problem with setup. *Computers & Operations Research*, 64:40 – 50, 2015.

[13] E. G. J. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer, 2013.

[14] F. Della Croce, F. Salassa, and R. Scatamacchia. An exact approach for the 0–1 knapsack problem with setups. *Computers & Operations Research*, 80:61–67, 2017.

[15] M. Dell'Amico, J. C. D. Díaz, and M. Iori. The bin packing problem with precedence constraints. *Operations Research*, 60(6):1491–1504, 2012.

[16] M. Dell'Amico, F. Furini, and M. Iori. A branch-and-price algorithm for the temporal bin packing problem. *Computers & Operations Research*, 114:104825, 2020.

[17] M. Delorme, M. Iori, and S. Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1 – 20, 2016.

[18] F. Focacci, F. Furini, V. Gabrel, D. Godard, and X. Shen. MIP formulations for a rich real-world lot-sizing problem with setup carryover. In R. Cerulli, S. Fujishige, and A. R. Mahjoub, editors, *ISCO 2016*, volume 9849 of *LNCS*, pages 123–134. Springer, 2016.

[19] F. Furini, M. Monaci, and E. Traversi. Exact approaches for the knapsack problem with setups. *Computers & Operations Research*, 90:208–220, 2018.

[20] M. Iori and S. Martello. Routing problems with loading constraints. *TOP*, 18(1):4–27, 2010.

[21] D. S. Johnson, A. J. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:299–325, 1974.

[22] E. Lin. A bibliographical survey on some well-known non-standard knapsack problems. *INFOR*, 36(4):274–317, 1998.

[23] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, New York, 1990.

[24] J. Martinovic, N. Strasdat, J. M. V. de Carvalho, and F. Furini. A combinatorial flow-based formulation for temporal bin packing problems. *European Journal of Operational Research*, 307 (2):554–574, 2023.

[25] S. Mehrani, C. Cardonha, and D. Bergman. Models and algorithms for the bin-packing problem with minimum color fragmentation. *INFORMS Journal on Computing*, 34(2):1070–1085, 2022.

[26] S. Michel, N. Perrot, and F. Vanderbeck. Knapsack problems with setups. *European Journal of Operational Research*, 196(3):909–918, 2009.

[27] U. Pferschy and R. Scatamacchia. Improved dynamic programming and approximation results for the knapsack problem with setups. *International Transactions in Operational Research*, 25 (2):667–682, 2018.

[28] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming.* Springer Publishing Company, Incorporated, 1st edition, 2010.

[29] H. Pollaris, K. Braekers, A. Caris, G. K. Janssens, and S. Limbourg. Vehicle routing problems with loading constraints: State-of-the-art and future directions. 37, 2015.

[30] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

[31] T. Sawik. *Scheduling in Supply Chains Using Mixed Integer Programming.* Wiley, 2011.

[32] P. Schwerin and G. Wäscher. The bin-packing problem: A problem generator and some numerical experiments with ffd packing and mtp. *International Transactions in Operational Research*, 4(5):377–389, 1997.

[33] H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling*, 4(6):313 – 338, 2001.

[34] L. Wei, Z. Luo, R. Baldacci, and A. Lim. A New Branch-and-Price-and-Cut Algorithm for One-Dimensional Bin-Packing Problems. *INFORMS Journal on Computing*, 32(2):428–443, 2020.

[35] E. Xavier and F. Miyazawa. The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science*, 393(1):240–259, 2008.

[36] Y. Yang and R. Bulfin. An exact algorithm for the knapsack problem with setup. *International Journal of Operational Research*, 5(3):280–291, 2009.

Electronic Companion

4.1. Notation summary

In Table 3, we summarize all the notation used throughout the paper. The first block lists the data and parameters of the classical BPP, while the second block introduces the additional data required for the BPPS. The third block reports auxiliary parameters and valid bounds that will be used in the analysis of our formulations. The fourth block defines the decision variables employed in the ILP models. The last two blocks describe the natural ILP formulation ILP_N and the three proposed enhancement ($\text{ILP}_\text{N}^\dagger$, $\text{ILP}_\text{N}^\ddagger$ and $\text{ILP}_\text{N}^\star$) together with their LP relaxations, whose optimal values provide lower bounds on ψ.

4.2. A structural property of the BPP

An instance of the BPP is defined by a bin capacity d and a set of items $\mathcal{I} = \{1, 2, \dots, n\}$, where each item $i \in \mathcal{I}$ has a positive integer weight $w_i \in \mathbb{Z}_{>0}$. The minimum number of bins required to pack all items of a BPP instance is denoted by β. An instance $I(BPP)$ of the BPP is defined by the tuple $(n, \boldsymbol{w}, d)$, where $\boldsymbol{w} \in \mathbb{Z}^n_{>0}$ is the vector of item weights. In the literature (see, e.g., [23, Section 8.3.2]), the following inequality is known to hold for every BPP instance:

$$\left\lceil \frac{\sum_{i\in\mathcal{I}} w_i}{d} \right\rceil \ \geq \ \frac{\beta}{2}, \tag{12}$$

which states that the ceiling of the total weight of the items divided by the bin capacity is always at least half the minimum number of bins. The next lemma provides a slight generalization of this result, used in the proof of Proposition 5, showing that, whenever more than one bin is required in any optimal BPP solution, the total weight of the items must exceed half of the total capacity of the bins in that solution.

Lemma 1. *For every BPP instance, if $\beta > 1$ then*

$$\sum_{i\in\mathcal{I}} w_i \ > \ \frac{\beta\, d}{2}. \tag{13}$$

Proof. Consider any optimal solution to the BPP. Let $\bar{\mathcal{B}}$ be the set of the β bins used in this optimal solution and, for every $b \in \bar{\mathcal{B}}$, let $\mathcal{I}_b(\bar{\mathcal{B}})$ be the set of items packed in bin b. In an optimal solution, no two bins can have a total item weight at most $\frac{d}{2}$ since, otherwise, they could be merged into a single bin. If all bins have load greater than $\frac{d}{2}$, the claim follows immediately since

$$\sum_{i\in\mathcal{I}} w_i \ = \sum_{b\in\bar{\mathcal{B}}} \sum_{i\in\mathcal{I}_b(\bar{\mathcal{B}})} w_i \ > \ \frac{\beta\, d}{2}.$$

Table 3: Summary of notation

BPP givens	**Description**
$d \in \mathbb{Z}_{>0}$	Bin capacity
$n \in \mathbb{Z}_{>0}$	Number of items
$w_i \in \mathbb{Z}_{>0}$	Weight of item $i \in \mathcal{I}$
$I(BPP)$	Instance of the BPP
$\beta \in \mathbb{Z}_{>0}$	Optimal value for the BPP
BPPS givens	**Description**
$m \in \mathbb{Z}_{>0}$	Number of classes
$\mathcal{C} = \{1, 2, \ldots, m\}$	Set of item classes
$\mathcal{P} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m\}$	Partition of the items into classes
$\mathcal{I}_c \subseteq \mathcal{I}$	Subset of items belonging to class $c \in \mathcal{C}$
$s_c \in \mathbb{Z}_{\geq 0}$	Setup weight for class $c \in \mathcal{C}$
$f_c \in \mathbb{Z}_{\geq 0}$	Setup cost for class $c \in \mathcal{C}$
$r \in \mathbb{Z}_{>0}$	Bin cost
$I(BPPS)$	Instance of the BPPS
$\psi \in \mathbb{Z}_{>0}$	Optimal value for the BPPS
BPPS additional givens	**Description**
β_c	Optimal value for the BPP with items of class $c \in \mathcal{C}$ and bin capacity $d - s_c$
$\overline{\beta}_c$	Upper bound on the number of bins to pack all items $i \in \mathcal{I}_c$ and bin capacity $d - s_c$
$\gamma_c = \lceil \sum_{i \in \mathcal{I}_c} w_i \, / (d - s_c) \rceil$	Valid lower bound on the number of bins required for class $c \in \mathcal{C}$
k	Upper bound on the number of bins used in any optimal BPPS solution
$\mathcal{B} = \{1, 2, \ldots, k\}$	Set of available bins
$\overline{k} = \sum_{c \in \mathcal{C}} \overline{\beta}_c$	Valid upper bound on the number of bins used in any optimal BPPS solution
$\underline{k} = \lceil (\sum_{i \in \mathcal{I}} w_i \; + \; \sum_{c \in \mathcal{C}} \gamma_c \, s_c)/d \rceil$	Lower bound on the number of bins used in any optimal BPPS solution
ILP formulations for the BPPS	**Description**
$\mathrm{ILP_N}$	Natural formulation for the BPPS
$\mathrm{ILP_N^\dagger}$	$\mathrm{ILP_N}$ with MCIs
$\mathrm{ILP_N^\ddagger}$	$\mathrm{ILP_N}$ with MCIs and MBI
$\mathrm{ILP_N^\star}$	$\mathrm{ILP_N}$ with MCIs and MBI and UB to the number of bins in any optimal solution
Lower bounds on ψ	**Description**
$\zeta(\mathrm{LP_N})$, $\zeta(\mathrm{LP_N^\dagger})$, $\zeta(\mathrm{LP_N^\ddagger})$, $\zeta(\mathrm{LP_N^\star})$	Optimal value of the LP relaxation of $\mathrm{ILP_N}$, $\mathrm{ILP_N^\dagger}$, $\mathrm{ILP_N^\ddagger}$ and $\mathrm{ILP_N^\star}$, respectively

Otherwise, there is at most one bin $\underline{b} \in \bar{\mathcal{B}}$ with total load $\sum_{i\in\mathcal{I}_{\underline{b}}} w_i \le \frac{d}{2}$. Since $\beta > 1$, there exists another bin $\bar{b} \in \bar{\mathcal{B}}$; by optimality, we must have

$$\sum_{i\in\mathcal{I}_{\bar{b}}(\bar{\mathcal{B}})} w_i + \sum_{i\in\mathcal{I}_{\underline{b}}(\bar{\mathcal{B}})} w_i \;>\; d,$$

otherwise the two bins $\bar{b}$ and $\underline{b}$ could be merged.

Hence,

$$\sum_{i\in\mathcal{I}} w_i = \sum_{\substack{i\in\mathcal{I}:\\ i\notin\mathcal{I}_{\underline{b}}(\bar{\mathcal{B}})\cup\mathcal{I}_{\bar{b}}(\bar{\mathcal{B}})}} w_i + \sum_{i\in\mathcal{I}_{\bar{b}}(\bar{\mathcal{B}})} w_i + \sum_{i\in\mathcal{I}_{\underline{b}}(\bar{\mathcal{B}})} w_i \;>\; \frac{d}{2}(\beta-2) + d = \frac{\beta\, d}{2}.$$

□

4.3. Proofs

4.3.1. Proof of Proposition 1

Proof. Multiplying the capacity constraints (1c) by the positive bin cost r and summing over all $b \in \mathcal{B}$ gives

$$r\sum_{i\in\mathcal{I}} w_i \sum_{b\in\mathcal{B}} x_{ib} + r\sum_{c\in\mathcal{C}} s_c \sum_{b\in\mathcal{B}} y_{cb} \;\le\; d\sum_{b\in\mathcal{B}} r\, z_b.$$

By the assignment constraints (1b), $\sum_{b\in\mathcal{B}} x_{ib} = 1$ for every $i \in \mathcal{I}$. Since every class $c \in \mathcal{C}$ contains at least one item $i \in \mathcal{I}_c$, the linking constraints (1d) imply

$$\sum_{b\in\mathcal{B}} y_{cb} \;\ge\; \sum_{b\in\mathcal{B}} x_{ib} \;=\; 1. \tag{14}$$

Therefore

$$r\sum_{i\in\mathcal{I}} w_i + r\sum_{c\in\mathcal{C}} s_c \;\le\; d\sum_{b\in\mathcal{B}} r\, z_b, \quad\Longrightarrow\quad \sum_{b\in\mathcal{B}} r\, z_b \;\ge\; \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right). \tag{15}$$

Moreover,

$$\sum_{b\in\mathcal{B}}\sum_{c\in\mathcal{C}} f_c\, y_{cb} \;=\; \sum_{c\in\mathcal{C}} f_c \sum_{b\in\mathcal{B}} y_{cb} \;\ge\; \sum_{c\in\mathcal{C}} f_c.$$

Therefore, every feasible solution of $\mathrm{LP_N}$ satisfies

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) \;\ge\; \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c \tag{16}$$

and, accordingly, the right-hand side of (16) is a lower bound on $\zeta(\mathrm{LP_N})$. Now consider the solution defined in (3). It satisfies constraints (2) by construction. Indeed, for every $i \in \mathcal{I}$,

$$\sum_{b\in\mathcal{B}} x_{ib} = \underbrace{k\,\frac{1}{k}}_{\text{by (3a)}} = 1,$$

so the assignment constraints (1b) hold. Moreover, for each $b \in \mathcal{B}$,

$$\sum_{i\in\mathcal{I}} w_i\, x_{ib} + \sum_{c\in\mathcal{C}} s_c\, y_{cb} = \underbrace{\sum_{i\in\mathcal{I}} w_i\,\frac{1}{k}}_{\text{by (3a)}} + \underbrace{\sum_{c\in\mathcal{C}} s_c\,\frac{1}{k}}_{\text{by (3b)}} = \frac{1}{k}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) = \underbrace{d\, z_b}_{\text{by (3c)}},$$

so the capacity constraints (1c) also hold at equality. Finally, the linking constraints (1d) are satisfied since by (3a) and (3b) the x-variables and y-variables take the same value $1/k$.

Hence, the solution defined in (3) is a feasible solution for $\mathrm{LP_N}$. Its objective value is

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) = \underbrace{r\sum_{b\in\mathcal{B}} \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c}{k\, d}}_{\text{by (3c)}} + \underbrace{\sum_{c\in\mathcal{C}} f_c \sum_{b\in\mathcal{B}} \frac{1}{k}}_{\text{by (3b)}} = \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c.$$

Thus, the objective value of the solution in (3) coincides with the right-hand side of (16). Consequently, the solution attains the lower bound and is therefore optimal, with optimal value as in (4). □

4.3.2. Proof of Proposition 3

Proof. Consider a feasible solution to $\mathrm{ILP_N}$. From the assignment constraints (1b) and the linking constraints (1d) we know that, for every $c \in \mathcal{C}$, if an item of class c is packed in bin b, then $y_{cb} = 1$. Thus, items of any class $c \in \mathcal{C}$ can only be packed in bins b for which $y_{cb} = 1$. For every $c \in \mathcal{C}$, let us denote this set of bins as

$$\mathcal{B}_c = \{\, b \in \mathcal{B} \mid y_{cb} = 1 \,\}.$$

For every $b \in \mathcal{B}_c$, the capacity constraints (1c) imply that

$$\underbrace{\sum_{i\in\mathcal{I}} w_i\, x_{ib}}_{\geq \sum_{i\in\mathcal{I}_c} w_i\, x_{ib}} + s_c + \underbrace{\sum_{g\in\mathcal{C},\, g\neq c} s_g\, y_{gb}}_{\geq 0} \;\leq\; d \quad\Longrightarrow\quad \sum_{i\in\mathcal{I}_c} w_i\, x_{ib} \;\leq\; d - s_c.$$

Summing over all $b \in \mathcal{B}_c$ yields

$$\sum_{b\in\mathcal{B}_c} \sum_{i\in\mathcal{I}_c} w_i\, x_{ib} \;\leq\; \sum_{b\in\mathcal{B}_c} (d - s_c).$$

The left-hand side equals $\sum_{i\in\mathcal{I}_c} w_i$, since items $i \in \mathcal{I}_c$ are packed in bins in $\mathcal{B}_c$, and thus $\sum_{b\in\mathcal{B}_c} x_{ib} = \sum_{b\in\mathcal{B}} x_{ib} = 1$. The right-hand side equals $|\mathcal{B}_c|(d - s_c) = \left(\sum_{b\in\mathcal{B}} y_{cb}\right)(d - s_c)$, since $y_{cb} = 1$ only if $b \in \mathcal{B}_c$. Therefore, for every $c \in \mathcal{C}$, we have

$$\sum_{i\in\mathcal{I}_c} w_i \;\le\; \left(\sum_{b\in\mathcal{B}} y_{cb}\right)(d - s_c) \quad\Longrightarrow\quad \sum_{b\in\mathcal{B}} y_{cb} \;\ge\; \frac{\sum_{i\in\mathcal{I}_c} w_i}{d - s_c}.$$

Since $\sum_{b\in\mathcal{B}} y_{cb}$ is an integer, the right-hand side can be rounded up, which coincides with the definition of γ_c. □

4.3.3. Proof of Proposition 4

Proof. The proof follows the same logical scheme as that of Proposition 1. The only difference is that, instead of using (14), we now directly rely on the MCI (5). It follows that every feasible solution of $\mathrm{LP}_\mathrm{N}^\dagger$ satisfies

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) \;\ge\; \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) \;+\; \sum_{c\in\mathcal{C}} \gamma_c\, f_c. \tag{17}$$

and, accordingly, the right-hand side of (17) is a lower bound on $\zeta(\mathrm{LP}_\mathrm{N}^\dagger)$. Now consider the solution defined in (6). As in Proposition 1, it can be verified that this solution satisfies all the constraints of $\mathrm{LP}_\mathrm{N}^\dagger$, and is therefore a feasible solution. Its objective value is

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) = \underbrace{r\sum_{b\in\mathcal{B}} \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{k\, d}}_{\text{by (6c)}} + \underbrace{\sum_{c\in\mathcal{C}} f_c \sum_{b\in\mathcal{B}} \frac{\gamma_c}{k}}_{\text{by (6b)}}$$

$$= \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) + \sum_{c\in\mathcal{C}} \gamma_c\, f_c.$$

Thus, the objective value of this solution coincides with the right-hand side of (17). Consequently, the solution attains the lower bound and is therefore optimal, with optimal value (7). □

4.3.4. Proof of Proposition 5

Proof. The proof relies on determining a feasible BPPS solution, obtained through the following three steps. We refer to this procedure as the *Constructive Heuristic Algorithm* (CHA) for the BPPS. At the end of each step, the algorithm could provide a heuristic solution whose half value is less than $\zeta(\mathrm{LP}_\mathrm{N}^\dagger)$. If not, we continue improving the heuristic solution in the next step by merging bins, until the inequality we seek holds. Depending on the case, it may be easier to show the inequality, using a worse heuristic solution, i.e., one with a higher value. This is why we make the CHA stop after only the first or the second step, depending on the case.

Having an heuristic solution whose half value is less than $\zeta(\mathrm{LP}^{\dagger}_{\mathrm{N}})$ is enough to prove the proposition. Indeed, the thesis of the proposition is equivalent to

$$\zeta(\mathrm{LP}^{\dagger}_{\mathrm{N}}) > \frac{1}{2}\psi \tag{18}$$

and since any heuristic solution value $\overline{\psi}$ is such that $\overline{\psi} \geq \psi$, to prove (18) it is enough to show that $\zeta(\mathrm{LP}^{\dagger}_{\mathrm{N}}) > \frac{1}{2}\overline{\psi}$, for any heuristic solution value $\overline{\psi}$.

These are the three steps of the CHA:

1. In the first step of the CHA, for each class $c \in \mathcal{C}$, we define an associated BPP instance obtained by restricting the item set to $\mathcal{I}_c$ with weights $\{w_i\}_{i\in\mathcal{I}_c}$, and by setting the bin capacity to $d_c = d - s_c$, i.e., the residual capacity after accounting for the setup weight of class c. This BPP instance, denoted by $I_c(BPP)$, is represented by the tuple:

   $$\underbrace{\left(|\mathcal{I}_c|,\, \{w_i\}_{i\in\mathcal{I}_c},\, d_c\right)}_{=I_c(BPP)}.$$

   We assume that all these m BPP instances, one for each class, are solved to optimality.

   We denote by β_c the corresponding optimal value, i.e., the minimum number of bins of capacity d_c required to pack all the items of class c. We further denote by

   $$\mathcal{B}_c = \{1, 2, \ldots, \beta_c\}$$

   the set of bins in an optimal solution of instance $I_c(BPP)$, and by $\mathcal{I}_b(\mathcal{B}_c)$ the set of items packed in each bin $b \in \mathcal{B}_c$. Finally, let $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ denote the subset of classes (if any) whose items can be packed into a single bin, i.e., for which $\beta_c = 1$. The outcome of this step is therefore the creation of bins that contain only items belonging to the same class, with the items of each class $c \in \mathcal{C}$ partitioned into β_c such bins, together with the identification of the subset $\tilde{\mathcal{C}}$ of classes whose items fit entirely in a single bin. If $\tilde{\mathcal{C}} = \emptyset$, i.e., when every class requires at least two bins, the CHA terminates at this point; otherwise, it proceeds to Step 2.

   At the end of this step, since we solved the BPPs with residual capacities, we obtain a feasible solution for the BPPS which uses $\sum_{c\in\mathcal{C}} \beta_c$ bins, where items of any class c are packed in exactly β_c bins. Hence, this feasible solution has a value of

   $$\overline{\psi}_1(\mathrm{CHA}) := \sum_{c\in\mathcal{C}} \beta_c f_c + r \sum_{c\in\mathcal{C}} \beta_c. \tag{19}$$

2. In the second step of the CHA, we define an additional BPP instance in which the items correspond to the classes in $\tilde{\mathcal{C}}$.

   Specifically, each class $c \in \tilde{\mathcal{C}}$ is represented by a single *class-aggregated item* of weight $\sum_{i\in\mathcal{I}_c} w_i + s_c$, while the bin capacity remains d. This BPP instance, denoted by $I_{\tilde{\mathcal{C}}}(BPP)$, is represented by the tuple:

$$\underbrace{\big(|\tilde{\mathcal{C}}|,\, \{\sum_{i\in\mathcal{I}_c} w_i + s_c\}_{c\in\tilde{\mathcal{C}}},\, d\big)}_{=I_{\tilde{\mathcal{C}}}(BPP)}.$$

   We assume that this BPP instance is solved to optimality, and we denote by δ the corresponding optimal value, i.e., the minimum number of bins of capacity d required to pack all the class-aggregated items. We further denote by

$$\mathcal{B}_{\tilde{\mathcal{C}}} = \{1, 2, \ldots, \delta\}$$

   the set of bins in an optimal solution of instance $I_{\tilde{\mathcal{C}}}(BPP)$, and by $\mathcal{I}_b(\mathcal{B}_{\tilde{\mathcal{C}}})$ the aggregated items packed in each bin $b \in \mathcal{B}_{\tilde{\mathcal{C}}}$. If $\delta \geq 2$, i.e., when the class-aggregated items cannot all be packed into a single bin of capacity d, the outcome of this step is the creation of bins that contain items of different classes obtained by merging some of the bins obtained in Step 1. In this case, when $\delta \geq 2$, the CHA terminates at this point; otherwise, it proceeds to Step 3.

   At the end of this step, since we just merged some bins if the capacities and setup cost allowed it, we obtain a feasible solution for the BPPS which uses $\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c + \delta$ bins, and items of any class c are still packed in exactly β_c bins. Hence this feasible solution has a value of

$$\overline{\psi}_2(\text{CHA}) := \sum_{c\in\mathcal{C}} \beta_c f_c + r\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c + \delta\right). \qquad (20)$$

3. In the third step of the CHA, we create a single meta item of weight $\sum_{c\in\tilde{\mathcal{C}}}\big(\sum_{i\in\mathcal{I}_c} w_i + s_c\big)$, which corresponds to the total weight of all items belonging to the classes in $\tilde{\mathcal{C}}$ plus the setup weight of each such class. This is the total capacity occupied in the unique new bin created in Step 2. Next, we select any class $\bar{c} \in \mathcal{C} \setminus \tilde{\mathcal{C}}$. It is worth noting that such a class exists since if $\mathcal{C} \setminus \tilde{\mathcal{C}} = \emptyset$ after step 1 and $\delta = 1$, then we are in the situation in which all items $i \in \mathcal{I}$ fit in a single bin, which is a trivial case that we did not consider in this paper, as mentioned in the Introduction. We then try to place the newly created meta item into the residual capacity of one of the bins in $\mathcal{B}_{\bar{c}}$. In this way, the bin created in Step 2—containing all items from $\tilde{\mathcal{C}}$—may be merged with one of the bins obtained in Step 1 or not.

The process just described corresponds to solving an additional BPP instance, in which the item set consists of the meta item associated with $\tilde{\mathcal{C}}$ together with the bins of class $\bar{c}$, each bin $b \in \mathcal{B}_{\bar{c}}$ being represented by a single item of weight $\sum_{i\in\mathcal{I}_b(\mathcal{B}_{\bar{c}})} w_i$, while the bin capacity is $d - s_{\bar{c}}$.

This additional BPP instance, denoted by $I_{\bar{c}}(BPP)$, is therefore represented by the tuple:

$$\underbrace{\big(1+\beta_{\bar{c}},\, \{\sum_{c\in\tilde{\mathcal{C}}}(\sum_{i\in\mathcal{I}_c} w_i + s_c)\} \cup \{\sum_{i\in\mathcal{I}_b(\mathcal{B}_{\bar{c}})} w_i\}_{b\in\mathcal{B}_{\bar{c}}},\, d-s_{\bar{c}}\big)}_{=I_{\bar{c}}(BPP)}.$$

At the end of this step, there are two possible outcomes, which both terminate the CHA. If the single meta item fits in one of the bins containing the items of class $\bar{c}$, we obtain a feasible solution for the BPPS which uses $\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c$ bins, and items of any class c are still packed in exactly β_c bins. Hence this feasible solution has a value of

$$\overline{\psi}_3(\text{CHA}) := \sum_{c\in\mathcal{C}} \beta_c f_c + r \sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c. \tag{21}$$

On the other hand, if the single meta item does not fit in one of the bins containing the items of class $\bar{c}$, then the feasible solution remains unchanged from the one of Step 2 and we do not improve our value which remains $\overline{\psi}_2(\text{CHA})$ which is in this case, with $\delta = 1$, equal to

$$\sum_{c\in\mathcal{C}} \beta_c f_c + r \left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c \,+\, 1 \right). \tag{22}$$

We want now to show, that at all the different terminations of the CHA the half value of the heuristic solution obtain is less than $\zeta(\text{LP}^\dagger_\text{N})$, which is saying that $\zeta(\text{LP}^\dagger_\text{N})$ is more than half of (19) or (20) or (21) or (22), depending on the respective termination of the CHA.

Let's consider $\zeta(\text{LP}^\dagger_\text{N})$, computed as in (7):

$$\zeta(\text{LP}^\dagger_\text{N}) \;=\; \sum_{c\in\mathcal{C}} \gamma_c\, f_c + \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right)$$

which has a component of the form $\sum_{c\in\mathcal{C}} \gamma_c f_c$. We observe that all (19), (20), (21), (22) have a component of the form $\sum_{c\in\mathcal{C}} \beta_c f_c$, and we have

$$\sum_{c\in\mathcal{C}} \gamma_c\, f_c \geq \frac{\sum_{c\in\mathcal{C}} \beta_c\, f_c}{2}.$$

because of (12) applied to the BPP instance $I_c(BPP)$, for every $c \in \mathcal{C}$.

Hence, to prove inequality that $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger}) > \frac{1}{2}$ is more than half of the value of the heuristic solution obtained with the CHA, it remains to show, depending on the termination of the CHA, the following four inequalities:

- $\frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \frac{1}{2}\, r \sum_{c\in\mathcal{C}} \beta_c$ if the CHA terminates at the end of Step 1, which means that $\beta_c \geq 2$ for every $c \in \mathcal{C}$ and $\tilde{\mathcal{C}}$ is empty. This inequality can be shown observing that

  $$\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\; s_c\right) = \frac{1}{d}\sum_{c\in\mathcal{C}}\left(\sum_{i\in\mathcal{I}_c} w_i + \gamma_c s_c\right) > \frac{1}{2}\sum_{c\in\mathcal{C}} \beta_c.$$

  Indeed, for every $c \in \mathcal{C}\setminus\tilde{\mathcal{C}}$, which includes all classes $c \in \mathcal{C}$ in this specific termination, we have

  $$\sum_{i\in\mathcal{I}_c} w_i + \gamma_c\, s_c \geq \sum_{i\in\mathcal{I}_c}\left(w_i + \frac{w_i}{d-s_c}\; s_c\right) = \sum_{i\in\mathcal{I}_c}\left(\frac{(d-s_c)\, w_i + s_c\, w_i}{d-s_c}\right)$$

  $$= d\,\frac{\sum_{i\in\mathcal{I}_c} w_i}{d-s_c} > \frac{\beta_c\, d}{2} \tag{23}$$

  where the last inequality comes from Lemma 1, applied to the BPP instance $I_c(BPP)$, which has an optimal solution using at least two bins since $c \in \mathcal{C}\setminus\tilde{\mathcal{C}}$.

- $\frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \frac{1}{2}\, r\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c\, +\, \delta\right)$ if the CHA terminates at the end of Step 2, which means that $\delta \geq 2$. This inequality can be shown observing that

  $$\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\; s_c\right) = \frac{1}{d}\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + \gamma_c s_c\right) + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right)\right)$$

  $$> \frac{1}{2}\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c\, +\, \frac{1}{2}\,\delta.$$

  where, for every $c \in \mathcal{C}\setminus\tilde{\mathcal{C}}$, we used (23), and for every $c \in \tilde{\mathcal{C}}$, we used

  $$\sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right) > \frac{\delta\, d}{2} \tag{24}$$

  obtained by applying Lemma 1 to the BPP instance solved in step 2, which has an optimal solution using at least two bins when the CHA terminates at the end of Step 2, since $\delta > 1$.

- $\frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \frac{1}{2}\, r \sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c$ if the CHA terminates at the end of Step 3 and the single meta item fits in one of the bins where the items of some class $\bar{c} \in \mathcal{C}\setminus\tilde{\mathcal{C}}$ are packed. This inequality can be shown observing that

$$\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) = \frac{1}{d}\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + \gamma_c s_c\right) + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right)\right)$$

$$> \frac{1}{2}\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c$$

where, for every $c \in \mathcal{C}\setminus\tilde{\mathcal{C}}$, we used (23).

- $\frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \frac{1}{2}\, r\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c\, + 1\right)$ if the CHA terminates at the end of Step 3 and the single meta item does not fit in one of the bins where the items of some class $\bar{c} \in \mathcal{C}\setminus\tilde{\mathcal{C}}$ are packed. This inequality can be shown observing that

$$\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) = \frac{1}{d}\left(\sum_{\substack{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}\\ c\neq\bar{c}}}\left(\sum_{i\in\mathcal{I}_c} w_i + \gamma_c s_c\right) + \sum_{i\in\mathcal{I}_{\bar{c}}} w_i + \gamma_{\bar{c}} s_{\bar{c}} + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right)\right)$$

$$> \frac{1}{2}\sum_{\substack{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}\\ c\neq\bar{c}}} \beta_c\, + \frac{1}{2}\,(\beta_{\bar{c}} + 1) = \frac{1}{2}\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c\, + 1\right). \tag{25}$$

where, for every $c \in \mathcal{C}\setminus\tilde{\mathcal{C}}$ different from $\bar{c}$, we used (23). Moreover, we have

$$\sum_{i\in\mathcal{I}_{\bar{c}}} w_i + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right) > (d - s_c)\frac{\beta_{\bar{c}} + 1}{2} \tag{26}$$

by applying Lemma 1 to the BPP instance $I_{\bar{c}}(BPP)$ solved in step 3 which has an optimal solution using at least two bins since in this case no bins were joined. From (26) we obtain

$$d\frac{\beta_{\bar{c}} + 1}{2} < \sum_{i\in\mathcal{I}_{\bar{c}}} w_i + s_c\frac{\beta_{\bar{c}} + 1}{2} + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right).$$

Hence, to show the inequality in (25) it remains only to show that

$$\gamma_{\bar{c}} \geq \frac{\beta_{\bar{c}} + 1}{2}$$

which is true because $\gamma_{\bar{c}} = \left\lceil \frac{\sum_{i\in\mathcal{I}_{\bar{c}}} w_i}{d - s_c} \right\rceil$, and $\frac{\sum_{i\in\mathcal{I}_{\bar{c}}} w_i}{d - s_c} > \frac{\beta_{\bar{c}}}{2}$ (by applying Lemma 1 to the BPP$_{\bar{c}}$ instance, since $\bar{c} \in \mathcal{C}\setminus\tilde{\mathcal{C}}$) and because $\beta_{\bar{c}}$ is also an integer.

□

4.3.5. *Proof of Proposition 7*

Proof. By inequality (17) in the proof of Proposition 4, every feasible solution of $\mathrm{LP}^{\dagger}_{\mathrm{N}}$—and hence of $\mathrm{ILP}_{\mathrm{N}}$—satisfies

$$\sum_{b\in\mathcal{B}}\Big(r\,z_b+\sum_{c\in\mathcal{C}} f_c\,y_{cb}\Big)\;\geq\;\frac{r}{d}\Big(\sum_{i\in\mathcal{I}} w_i+\sum_{c\in\mathcal{C}}\gamma_c s_c\Big)\;+\;\sum_{c\in\mathcal{C}}\gamma_c f_c.$$

By the MCI (5), we have $\sum_{b\in\mathcal{B}} y_{cb}\geq\gamma_c$ for all $c\in\mathcal{C}$, hence

$$\sum_{b\in\mathcal{B}}\sum_{c\in\mathcal{C}} f_c\,y_{cb}\;\geq\;\sum_{c\in\mathcal{C}}\gamma_c f_c.$$

Therefore,

$$r\sum_{b\in\mathcal{B}} z_b\;\geq\;\frac{r}{d}\Big(\sum_{i\in\mathcal{I}} w_i+\sum_{c\in\mathcal{C}}\gamma_c s_c\Big)\quad\Longrightarrow\quad\sum_{b\in\mathcal{B}} z_b\;\geq\;\frac{\sum_{i\in\mathcal{I}} w_i+\sum_{c\in\mathcal{C}}\gamma_c s_c}{d}.$$

Since the z-variables must take integer values in any feasible solution to $\mathrm{ILP}_{\mathrm{N}}$, we conclude that

$$\sum_{b\in\mathcal{B}} z_b\;\geq\;\left\lceil\frac{\sum_{i\in\mathcal{I}} w_i+\sum_{c\in\mathcal{C}}\gamma_c s_c}{d}\right\rceil.$$

□

4.3.6. *Proof of Proposition 8*

Proof. The proof follows the same logical scheme as that of Proposition 1. The only difference is that we now rely *directly* on the MBI (8) together with the MCI (5). Hence every feasible solution of $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$ satisfies

$$\sum_{b\in\mathcal{B}}\Big(r\,z_b+\sum_{c\in\mathcal{C}} f_c\,y_{cb}\Big)\;\geq\;r\sum_{b\in\mathcal{B}} z_b\;+\;\sum_{c\in\mathcal{C}} f_c\sum_{b\in\mathcal{B}} y_{cb}\;\geq\;r\,\underline{k}\;+\;\sum_{c\in\mathcal{C}}\gamma_c\,f_c, \tag{27}$$

Accordingly, the right-hand side of (27) is a lower bound on $\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}})$.

Now consider the point defined in (6). For each $b\in\mathcal{B}$,

$$\sum_{i\in\mathcal{I}} w_i\,x_{ib}+\sum_{c\in\mathcal{C}} s_c\,y_{cb}=\underbrace{\sum_{i\in\mathcal{I}} w_i\,\frac{1}{k}}_{\text{by (9a)}}+\underbrace{\sum_{c\in\mathcal{C}} s_c\,\frac{\gamma_c}{k}}_{\text{by (9b)}}=\frac{1}{k}\Big(\sum_{i\in\mathcal{I}} w_i+\sum_{c\in\mathcal{C}}\gamma_c s_c\Big)\;\leq\;\underbrace{\frac{d\,\underline{k}}{k}}_{\text{by (8)}}\;=\;\underbrace{d\,z_b}_{\text{by (9c)}},$$

so the capacity constraints (1c) are satisfied (in general, not at equality). The satisfaction of the other constraints is the same as in Proposition 1. Therefore, it can be verified that this point satisfies all the constraints of $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$, and is therefore a feasible solution. Its objective value is

$$\sum_{b\in\mathcal{B}}\Big(r\,z_b+\sum_{c\in\mathcal{C}} f_c\,y_{cb}\Big)=\underbrace{r\sum_{b\in\mathcal{B}}\frac{\underline{k}}{k}}_{\text{by (9c)}}\;+\;\underbrace{\sum_{c\in\mathcal{C}} f_c\sum_{b\in\mathcal{B}}\frac{\gamma_c}{k}}_{\text{by (9b)}}$$

$$= r\,\underline{k} + \sum_{c\in\mathcal{C}} \gamma_c\, f_c.$$

Thus, the objective value of this solution coincides with the right-hand side of (27). Consequently, the solution attains the lower bound and is therefore optimal, with optimal value (7). $\square$

4.3.7. *Proof of Proposition 9*

Proof. Consider the family of BPPS instances introduced in the proof of Proposition 6 where

$$\psi = n\,(r + f_1) \quad \text{and} \quad \gamma_1 \;=\; \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil.$$

By Proposition 8, the optimal value of $\mathrm{LP}^{\ddagger}_{\mathrm{N}}$is

$$\zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}}) = r \left\lceil \frac{n\,\vartheta + \left\lceil \frac{n\,\vartheta}{2\,\vartheta-1} \right\rceil}{2\,\vartheta} \right\rceil + \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil f_1 = r \left\lceil \frac{n}{2} + \frac{\left\lceil \frac{n\,\vartheta}{2\,\vartheta-1} \right\rceil}{2\,\vartheta} \right\rceil + \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil f_1$$

Taking the limit of $\zeta(\mathrm{LP}^{\dagger}_{\mathrm{N}})$ as $\vartheta \to \infty$ gives

$$\lim_{\vartheta\to\infty} \zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}}) = \lim_{\vartheta\to\infty} r \left\lceil \frac{n}{2} + \frac{\left\lceil \frac{n\,\vartheta}{2\,\vartheta-1} \right\rceil}{2\,\vartheta} \right\rceil + \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil f_1 = r \left\lceil \frac{n}{2} \right\rceil + \frac{n}{2}\, f_1\,.$$

Considering only even values for the number of items n, we therefore have that

$$\lim_{\vartheta\to\infty} \zeta(\mathrm{LP}^{\ddagger}_{\mathrm{N}}) = r\,\frac{n}{2} + \frac{n}{2}\, f_1 = \frac{n}{2}\,(r + f_1)\,,$$

which is exactly $\frac{1}{2}$ of the value of ψ for this class of instances. $\square$

4.3.8. *Proof of Proposition 10*

Proof. In any optimal solution to $\mathrm{ILP}_{\mathrm{N}}$ (or, equivalently, to $\mathrm{ILP}^{\dagger}_{\mathrm{N}}$), we have

$$\underbrace{r \sum_{b\in\mathcal{B}} z_b + \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}} f_c\, y_{cb}}_{=\psi} \;\le\; r \sum_{c\in\mathcal{C}} \beta_c + \sum_{c\in\mathcal{C}} f_c\, \beta_c, \tag{28}$$

since the optimal BPPS value ψ cannot exceed the value of the heuristic solution obtained in Step 1 of the Constructive Heuristic Algorithm described in the proof of Proposition 5.

Moreover, we have

$$r \sum_{c\in\mathcal{C}} \beta_c + \sum_{c\in\mathcal{C}} f_c\, \beta_c \;\le\; r \sum_{c\in\mathcal{C}} \beta_c + \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}} f_c\, y_{cb}, \tag{29}$$

because the items of any class $c \in \mathcal{C}$ cannot be packed into fewer than β_c bins. Indeed, for each class $c \in \mathcal{C}$ we have

$$f_c\,\beta_c \;\le\; f_c \sum_{b \in \mathcal{B}} y_{cb} \quad\Longrightarrow\quad \sum_{c \in \mathcal{C}} f_c\,\beta_c \;\le\; \sum_{b \in \mathcal{B}} \sum_{c \in \mathcal{C}} f_c\,y_{cb}.$$

By chaining inequalities (28) and (29), we obtain

$$r \sum_{b \in \mathcal{B}} z_b + \sum_{b \in \mathcal{B}} \sum_{c \in \mathcal{C}} f_c\,y_{cb} \;\le\; r \sum_{c \in \mathcal{C}} \beta_c + \sum_{b \in \mathcal{B}} \sum_{c \in \mathcal{C}} f_c\,y_{cb},$$

which implies

$$\sum_{b \in \mathcal{B}} z_b \;\le\; \sum_{c \in \mathcal{C}} \beta_c.$$

Therefore, the number of bins used in any optimal BPPS solution is at most equal to the sum, over all classes, of the minimum number of bins β_c required to pack the items of class $c \in \mathcal{C}$. In particular, replacing β_c with any valid upper bound $\overline{\beta}_c \ge \beta_c$ still yields a valid (though possibly weaker) overall bound:

$$\sum_{b \in \mathcal{B}} z_b \;\le\; \sum_{c \in \mathcal{C}} \overline{\beta}_c.$$

□