The Bin Packing Problem with Setups: Formulations, 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 the *Bin Packing Problem with Setups* (BPPS), a generalization of the classical Bin Packing Problem with applications in production planning and logistics. In this problem, the items are partitioned into classes, and packing items of a class in a bin incurs a setup weight and cost. We propose a natural *Integer Linear Programming* (ILP) formulation for the BPPS and analyze its *Linear Programming* relaxation. We show that the resulting lower bound can be arbitrarily weak and introduce the *Minimum Classes Inequalities* (MCIs), which guarantee a worst-case ratio of 1/2 with respect to the optimal objective function value of the BPPS. We also derive the *Minimum Bins Inequality* (MBI) and an upper bound on the number of bins in any optimal solution, substantially reducing the formulation size. We further develop an arc-flow formulation for the BPPS based on a tailored graph construction and compression procedure. Its LP relaxation dominates that of the natural formulation, and both the MCIs and the MBI are extended to the arc-flow model. Finally, we introduce a benchmark comprising 576 randomly generated instances and 36 real-world instances derived from a vehicle-routing application, and conduct extensive computational experiments. Results show that the natural formulation performs best on instances with small or medium item weights, whereas the arc-flow formulation is more effective for large item weights and on the real-world testbed.

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* $d \in \mathbb{Z}_{\geq 1}$, and a set $\mathcal{I} = \{1, 2, \ldots, n\}$ of n *items*, where each item $i \in \mathcal{I}$ has a positive integer *weight* $w_i \in \mathbb{Z}_{\geq 1}$, the *Bin Packing Problem* (BPP) consists of determining a partition of the items into the minimum number of *packing patterns*, where a packing pattern is a subset of items $S \subseteq \mathcal{I}$ whose total item weight 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. [2013] and Delorme et al. [2016] 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 $\mathcal{P} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m\}$, where, for each class $c \in \mathcal{C} = \{1, 2, \ldots, m\}$, $\mathcal{I}_c \subseteq \mathcal{I}$ 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* $r \in \mathbb{Z}_{\geq 1}$, which is incurred once for every used bin.

For any subset of items $S \subseteq \mathcal{I}$, let $\mathcal{C}(S) = \{c \in \mathcal{C} : S \cap \mathcal{I}_c \neq \emptyset\}$ denote its active classes. In the BPPS, a packing pattern is a subset $S \subseteq \mathcal{I}$ whose total item weight plus the setup weights of its active classes does not exceed the bin capacity:

$$\sum_{i \in S} w_i + \sum_{c \in \mathcal{C}(S)} s_c \leq d.$$

By construction, every packing pattern can be assigned to a single bin. A partition $\mathcal{S} \subseteq 2^{\mathcal{I}}$ of the

item set into packing patterns has cost

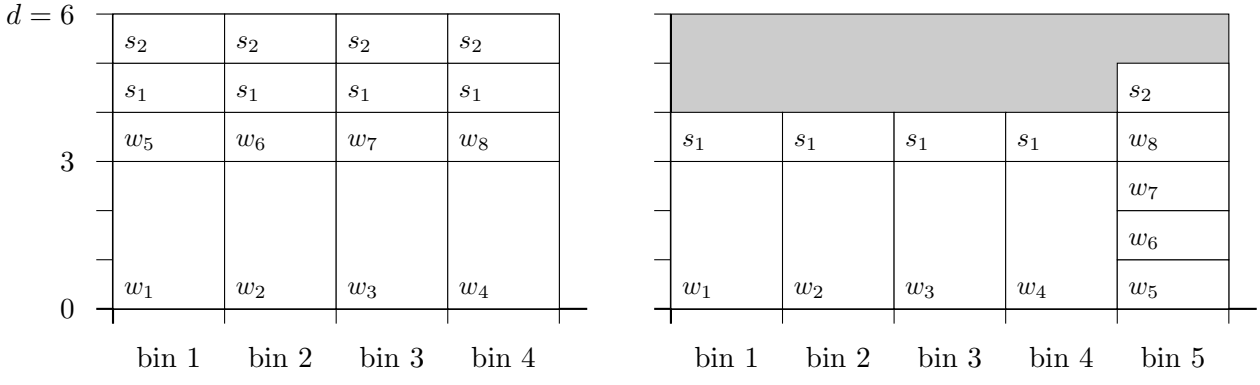
$$\sum_{S \in \mathcal{S}} \left(r + \sum_{c \in \mathcal{C}(S)} f_c \right) = r\,|\mathcal{S}| + \sum_{S \in \mathcal{S}} \sum_{c \in \mathcal{C}(S)} f_c, \tag{1}$$

that is, the sum, over all packing patterns $S \in \mathcal{S}$, of the cost of the bin containing the pattern plus the setup costs of the classes active in it. The BPPS consists in determining a partition into packing patterns of minimum cost. We denote by $\mathcal{S}^\star \subseteq 2^{\mathcal{I}}$ an optimal partition, and by OPT the corresponding optimal objective function value, i.e., the minimum cost required to pack all the items. The value of objective function (1) of any BPPS solution $\mathcal{S}$ provides an upper bound on the BPPS optimum, and is therefore denoted by $UB(\mathcal{S})$.

An instance of the BPPS is defined by the tuple $(n, \boldsymbol{w}, d, m, \mathcal{P}, \boldsymbol{s}, \boldsymbol{f}, r)$, where $\boldsymbol{w} \in \mathbb{Z}^n_{\geq 1}$ 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 the 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 $d - s_c$ and the item set corresponds to $\mathcal{I}_c$.

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 also implies that, unlike the classical BPP, where the number of used bins in an optimal solution coincides with the minimum number of bins required for packing all the items, in the BPPS using more bins than the minimum may lead to a better solution. This trade-off is illustrated by the two optimal BPPS solutions depicted in Figure 1.



(a) an optimal BPPS solution with $r = 10$ (b) an optimal BPPS solution with $r = 1$

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 space in each bin is shown in gray. In part (a), with $r = 10$, the optimal solution consists of four used bins: $\{\{1, 5\}, \{2, 6\}, \{3, 7\}, \{4, 8\}\}$, yielding a total minimum cost of $OPT = 60$. In part (b), with $r = 1$, the optimal solution consists of five used bins: $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5, 6, 7, 8\}\}$, with a total minimum cost of $OPT = 16$.

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 [2010] and Sawik [2011] 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 is 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, with each product requiring a given processing time (item weight) and belonging 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, for example, involve 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 [2015], where machinery requires setup operations when switching from one product type to another;

another example is 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. [2016].

Other important applications arise in logistics contexts where goods are distributed by vehicles that require special handling equipment, such as forklifts or conveyors, or by multi-compartment vehicles that transport different types of goods in separate compartments [Iori and Martello, 2010, Pollaris et al., 2015, Brecht et al., 2019]. 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 that the corresponding equipment be available on board, which reduces capacity and incurs equipment (setup) costs — e.g., from renting a forklift or installing specialized loading systems. Similarly, in the chemical and pharmaceutical industries [Bishara, 2006] and in the distribution of perishable goods [Brecht et al., 2019], certain products require refrigerated or frozen storage. 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 refrigeration energy consumption). In this paper, we consider a set of real-world instances of exactly this type, arising from the distribution of fresh and frozen grocery products by a major international retailer. In such scenarios, the BPPS models the problem of determining the optimal fleet composition and assigning customer orders to vehicles 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 [1994] and Lin [1998]. 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 [Akinc, 2006, Altay et al., 2008, Chebil and Khemakhem, 2015, Furini et al., 2018, Della Croce et al., 2017, Michel et al., 2009, Pferschy and Scatamacchia, 2018, Yang and Bulfin, 2009]. 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., [Baldacci et al., 2024, Ceselli and Righini, 2008, Dell'Amico et al., 2012, 2020, Martinovic et al., 2023, Sadykov and Vanderbeck, 2013, Wei et al., 2020]. Among the most relevant and related variants, the *Class-Constrained Bin Packing Problem* (CBPP) [Borges et al., 2020, Shachnai and Tamir, 2001, Xavier and Miyazawa, 2008] 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) [Barkel et al., 2025, Bergman et al., 2019, Casazza and Ceselli, 2014, Mehrani et al., 2022] 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 accounting for setup operations that are both capacity-consuming and cost-inducing, depending on the classes of items packed into each bin. Finally, it is worth mentioning the following two recent BPP generalizations. Baldi et al. [2019] introduce the *Generalized Bin Packing Problem with Bin-dependent Item Profits*, in which bins may have different costs and capacities, items may be compulsory or optional, and the profit generated by an item depends on the bin to which it is assigned. The objective is to minimize the cost of the selected bins minus the profits of the packed items. Crainic et al. [2021] study *Multi- and Single-Period Variable Cost and Size Bin Packing Problems with Assignment Costs*, which combine bin-selection costs with item-to-bin assignment costs; the multi-period variant also accounts for the availability of items and bins over time, while items that are not assigned to any bin incur an additional cost.

To position the BPPS within the nomenclature introduced by Crainic et al. [2021], we use their labeling scheme $D/C/B/K/T[\cdot]$. Here, D denotes the number of physical dimensions; $C \in \{U, V\}$ and $B \in \{U, V\}$ indicate whether bin costs and sizes, respectively, are unique or variable; $K \in \{S, M\}$ distinguishes between single- and multiple-type item settings; $T \in \{1, N, C\}$ denotes a single-period, multi-period, or continuous-time representation; and $[\cdot]$ lists additional problem-specific attributes. Accordingly, the BPPS can be labeled as $1/U/U/M = m/1$[Setup]: it is a one-dimensional, single-period problem with identical bins and m item classes, in which activating a class in a bin incurs both a setup cost and an additional capacity consumption. The attribute [Setup] identifies a mechanism not covered by the attributes explicitly listed in Crainic et al. [2021].

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 Classes Inequalities* (MCIs), that strengthen the LP relaxation of the natural ILP model. We also analyze this strengthened relaxation and derive closed-form expressions for its optimal solution. In Section 2.3, we prove that incorporating the MCIs yields a lower bound with worst-case performance ratio 1/2 relative to the optimal objective function value of the BPPS. 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 an optimal solution to 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 these theoretical results are provided in Section E.2 of the electronic companion due to space limitations. In Section 3, we present an *arc-flow formulation* for the BPPS, extending the construction of Brandão and Pedroso [2016] to handle item classes, setup weights, and setup costs. We show that the LP relaxation of this formulation dominates that of the natural ILP formulation, and we extend the MCIs and the MBI to this new formulation. Section 4 presents the computational study. Since the BPPS is a novel problem, we first introduce, in Section 4.1, a benchmark library of both real-world and synthetic instances that captures its main features, including variations in item weights, class numbers, setup costs, and setup weights. In Section 4.2, we assess the computational performance of both the natural and the arc-flow ILP formulations for the BPPS, evaluate the impact of the proposed enhancements on the effectiveness of such models, and compare the two formulation families against each other to identify the instance features that drive their performance. In Section 4.3, we assess the strength of the LP relaxations of the natural and arc-flow formulation families, confirming that the arc-flow relaxation is consistently tighter, provide insights into the main structural features of optimal BPPS solutions, and discuss why the stronger relaxation of the arc-flow formulation does not always translate into superior computational performance. The paper concludes with Section 5, 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 ILP formulation for the BPP (see, e.g., [Delorme et al., 2016]) 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 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 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. 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.

$$(\text{ILP}_\text{N}) \qquad \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \quad \sum_{b\in\mathcal{B}} \left(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb} \right) \tag{2a}$$

$$\sum_{b\in\mathcal{B}} x_{ib} = 1, \qquad i \in \mathcal{I}, \tag{2b}$$

$$\sum_{i\in\mathcal{I}} w_i\, x_{ib} + \sum_{c\in\mathcal{C}} s_c\, y_{cb} \leq d\, z_b, \qquad b \in \mathcal{B}, \tag{2c}$$

$$x_{ib} \leq y_{cb}, \qquad c \in \mathcal{C},\ i \in \mathcal{I}_c,\ b \in \mathcal{B}. \tag{2d}$$

The objective function (2a) 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* (2b) require each item to be packed into exactly one bin. The *capacity constraints* (2c) 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 are placed in an unused bin. For each bin $b \in \mathcal{B}$ and item $i \in \mathcal{I}$, the *linking constraints* (2d) 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 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 LP relaxation of $\mathrm{ILP_N}$, obtained by replacing the binary variables with the following ones:

$$0 \leq x_{ib} \leq 1,\ i \in \mathcal{I},\, b \in \mathcal{B}, \qquad 0 \leq y_{cb} \leq 1,\ c \in \mathcal{C},\, b \in \mathcal{B}, \qquad 0 \leq z_b \leq 1,\ b \in \mathcal{B}. \tag{3}$$

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

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

$$x_{ib} = \frac{1}{k},\ i \in \mathcal{I},\ b \in \mathcal{B}, \qquad y_{cb} = \frac{1}{k},\ c \in \mathcal{C},\ b \in \mathcal{B}, \qquad z_b = \frac{\sum_{i \in \mathcal{I}} w_i + \sum_{c \in \mathcal{C}} s_c}{k\, d},\ b \in \mathcal{B}, \tag{4}$$

and its optimal objective function value is

$$\zeta(\mathrm{LP_N}) = \frac{r}{d}\left(\sum_{i \in \mathcal{I}} w_i + \sum_{c \in \mathcal{C}} s_c\right) + \sum_{c \in \mathcal{C}} f_c. \tag{5}$$

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 objective function 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.

Following the notation used, e.g., in Martello and Toth [1990] and Delorme et al. [2016], let LB be a lower bound for a minimization problem. For any instance I of the problem, denote by $LB(I)$ the value of the lower bound LB and by $OPT(I)$ the optimal objective function value (assumed positive). The *worst-case performance ratio* of LB is defined as the largest real number $\varrho(LB) \in [0, 1]$ such that

$$\frac{LB(I)}{OPT(I)} \geq \varrho(LB) \quad \text{for all instances } I.$$

The following proposition shows that the worst-case performance ratio of the lower bound $\zeta(\mathrm{LP_N})$ (5) is zero and, consequently, this lower bound can be arbitrarily weak. The proof constructs a family of

instances parameterized by the number of items. In each instance, 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. In contrast, in an optimal fractional solution of $\mathrm{LP_N}$, both the setup and item weights can be fractionally distributed across all bins, yielding a much smaller optimal objective function value. The claim then follows by letting the number of items tend to infinity.

Proposition 2.

$$\varrho(\zeta(\mathrm{LP_N})) = 0.$$

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 $w_i = 1, i \in \mathcal{I}$, and $s_1 = n - 1$. 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 objective function value of the BPPS is $OPT = n\,(r + f_1)$. By Proposition 1, the optimal objective function value of $\mathrm{LP_N}$ is

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

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

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

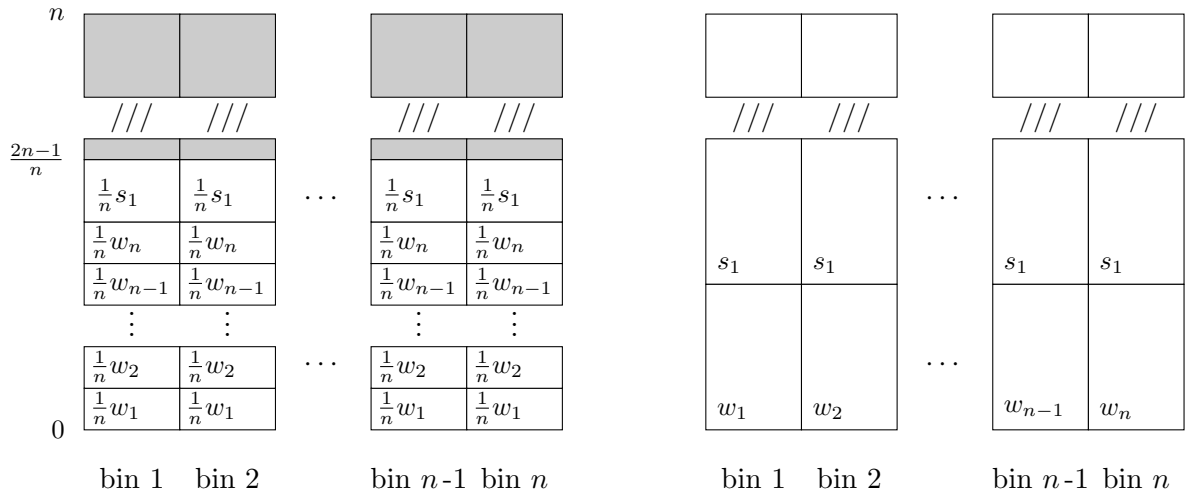
□

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.



(a) an optimal $\mathrm{LP_N}$ solution (b) an optimal BPPS solution

Figure 2: The left part of the figure illustrates an optimal solution to $\mathrm{LP_N}$ for the worst-case family of instances used in the proof of Proposition 2. 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 for the same instance, in which each bin of capacity n is completely filled by the weight of one item plus the class's setup weight.

2.2. Minimum Classes Inequalities

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

$$\sum_{b\in\mathcal{B}} y_{cb} \;\geq\; \underbrace{\left\lceil \frac{\sum_{i\in\mathcal{I}_c} w_i}{d-s_c} \right\rceil}_{=\gamma_c}, \qquad c\in\mathcal{C}, \tag{6}$$

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 in any optimal BPPS solution. 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 $\mathrm{LP_N}$ that would otherwise underestimate the actual packing requirements. The right-hand sides of the MCIs (6) 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 (6) are valid inequalities for $\mathrm{ILP_N}$.

Proposition 3. *For every BPPS instance, the Minimum Classes Inequalities* (6) *are satisfied by all feasible solutions of* $\mathrm{ILP_N}$.

The proof uses the constraints of $\mathrm{LP_N}$, 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 (6) to ILP_N, and by $\text{LP}_\text{N}^\dagger$ its LP relaxation. We let $\zeta(\text{LP}_\text{N}^\dagger)$ denote the optimal objective function value of $\text{LP}_\text{N}^\dagger$, which provides a lower bound on the optimal objective function value OPT 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 objective function value, as stated by the following proposition.

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

$$x_{ib} = \frac{1}{k},\ i \in \mathcal{I},\ b \in \mathcal{B}, \qquad y_{cb} = \frac{\gamma_c}{k},\ c \in \mathcal{C},\ b \in \mathcal{B}, \qquad z_b = \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{k\, d},\ b \in \mathcal{B}, \quad (7)$$

and its optimal objective function value is

$$\zeta(\text{LP}_\text{N}^\dagger) \ = \ \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) \ + \ \sum_{c\in\mathcal{C}} \gamma_c\, f_c. \quad (8)$$

Proposition 4 shows that an optimal solution to $\text{LP}_\text{N}^\dagger$ 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. These values are strictly larger than those in (4) whenever there exists a class $c \in \mathcal{C}$ such that $\gamma_c > 1$ and $s_c > 0$. The optimal objective function value of $\text{LP}_\text{N}^\dagger$ 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 $\text{ILP}_\text{N}^\dagger$

The inclusion of the MCIs (6) yields the following main theoretical result of the paper: the lower bound $\zeta(\text{LP}_\text{N}^\dagger)$ (8) is always strictly greater than one half of the optimal objective function value of the BPPS, OPT. This result shows that, unlike LP_N, the relaxation $\text{LP}_\text{N}^\dagger$ cannot yield arbitrarily weak lower bounds.

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

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

The proof of this proposition is based on the construction of a heuristic solution to the BPPS, obtained by solving several BPPs defined accounting for the setup costs of the classes in different ways: either by reducing the capacity, or by increasing the item weights, or by combining 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 because it requires a careful control of the heuristic solution's structure. In contrast, simpler constructions that would intuitively yield a better value do not make the result any easier to prove.

The following proposition shows that the worst-case performance ratio of the lower bound $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$ (8) is 1/2. The proof constructs a family of instances parameterized by the bin capacity, inspired by the classic worst-case examples for the BPP, see, e.g., [Martello and Toth, 1990, Section 8.3.2]. For these instances, the ratio $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})/OPT$ can be made arbitrarily close to 1/2. The claim then follows by letting the bin capacity tend to infinity.

Proposition 6.

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

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 $d = 2\,\vartheta$, $w_i = \vartheta, i \in \mathcal{I}$, $s_1 = 1$, and $f_1 = 0$. For each pair of distinct items $i, j \in \mathcal{I}$, we have that $w_i + w_j + s_1 = 2\,\vartheta + 1 > d$. Hence, each bin can contain at most one item. Therefore, any optimal BPPS solution uses one bin per item, and the optimal objective function value of the BPPS is $OPT = n\,r$. Moreover, we have $\gamma_1 = \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil$ and, by Proposition 4 and letting $k = n$, the optimal objective function value of $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ is

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

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

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

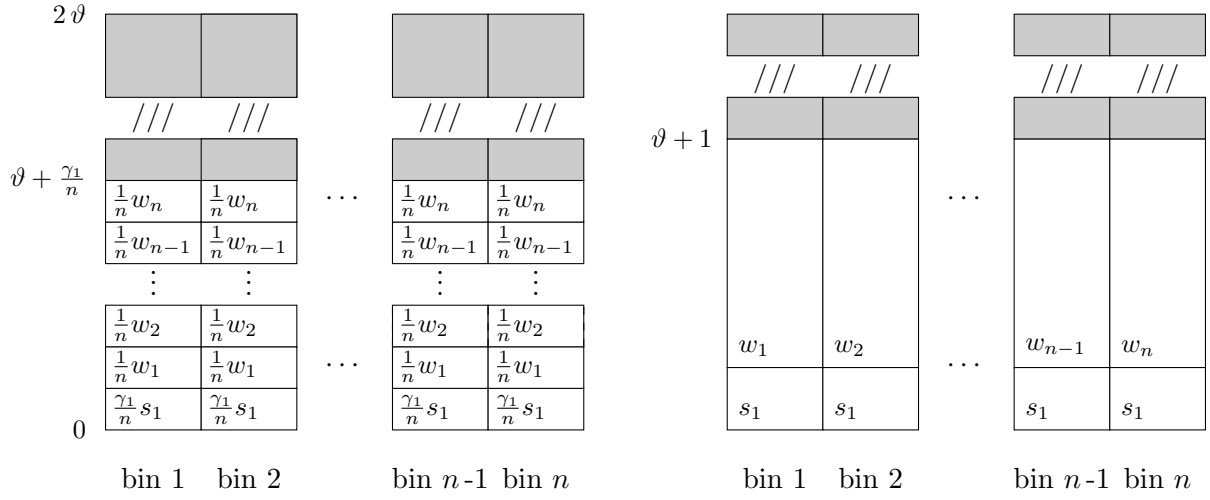
where the last equality follows from the fact that $\left\lceil \frac{n\vartheta}{2\vartheta - 1} \right\rceil \leq n$ for all $\vartheta \geq 1$, and hence $\frac{1}{2n\vartheta}\left\lceil \frac{n\vartheta}{2\vartheta - 1} \right\rceil \leq \frac{1}{2\vartheta} \to 0$ as $\vartheta \to \infty$. The result follows by combining the above limit with Proposition 5. □

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.



(a) an optimal $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ solution (b) an optimal BPPS solution

Figure 3: The left part of the figure illustrates an optimal solution to $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ for the worst-case family of instances used in the proof of Proposition 6. 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, in which at most one item can be packed in each bin of capacity 2 ϑ, since two items together with the setup weight do not fit.

Propositions 5 and 6 show that, with the inclusion of the MCIs, we can recover, for the BPPS, the classical result of a tight worst-case performance ratio of 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):

$$\sum_{b\in\mathcal{B}} z_b \;\geq\; \underbrace{\left\lceil \frac{\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{d} \right\rceil}_{=\check{k}}, \tag{9}$$

where the right-hand side $\check{k}$ 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 $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ that would otherwise underestimate the overall capacity requirement. The right-hand side of the MBI (9) 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 (9) is also a valid inequality for $\mathrm{ILP}_{\mathrm{N}}$.

Proposition 7. *For every BPPS instance, the Minimum Bins Inequality* (9) *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 $\mathrm{ILP}_{\mathrm{N}}^{\ddagger}$ the formulation obtained by adding the MCIs (6) and the MBI (9) to $\mathrm{ILP}_{\mathrm{N}}$, and by $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$ its LP relaxation. We let $\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger})$ denote the optimal objective function value of $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$, which provides a lower bound on the optimal objective function value OPT of the BPPS, and is at least as strong as $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$.

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

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

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

$$x_{ib} = \frac{1}{k},\ i \in \mathcal{I},\ b \in \mathcal{B}, \qquad y_{cb} = \frac{\gamma_c}{k},\ c \in \mathcal{C},\ b \in \mathcal{B}, \qquad z_b = \frac{\check{k}}{k},\ b \in \mathcal{B}, \tag{10}$$

and its optimal objective function value is

$$\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger}) \ = \ r\,\check{k} \ + \ \sum_{c \in \mathcal{C}} \gamma_c\, f_c. \tag{11}$$

While the MBI (9) provides a global lower bound on the total number of bins that must be used, the MCIs (6) 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 (9) constrains only the aggregate number of bins, without distinguishing which classes are responsible for their activation,

whereas the MCIs (6) ensure that the contribution of each class is explicitly enforced.

The following proposition shows that the worst-case performance ratio of the lower bound $\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger})$ (11) is 1/2, equal to that of $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$.

Proposition 9.

$$\varrho(\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger})) = \frac{1}{2}.$$

The proof employs the same instance of the BPPS used in Proposition 6, thereby showing the analogous result for the model $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$. In our computational experiments of Section 4, we empirically measure the actual improvement of the lower bound provided by $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ and $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$ 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 strongly impacts 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*

$$\hat{k} = \sum_{c \in \mathcal{C}} \overline{\beta}_c, \tag{12}$$

provides an 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 to the BPPS with the value of the first heuristic solution also considered in the proof of Proposition 5, which is obtained by solving a BPP 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 an 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 4.

3. An arc-flow ILP formulation for the BPPS

In this section, we introduce an arc-flow formulation for the BPPS by extending the approach of Brandão and Pedroso [2016]. Originally introduced for the BPP by Valério de Carvalho [1999], arc-flow formulations model feasible packing patterns as source-to-sink paths in a *directed acyclic graph* (DAG), using variables that represent flows on individual arcs of the DAG. We refer to de Lima et al. [2022] for a recent survey on the dynamic-programming foundations of arc-flow formulations.

In the classical bin packing setting, a path encodes a packing pattern whose total weight does not exceed the bin capacity. In the BPPS, however, the feasibility and cost of a packing pattern also depend on the classes of the items it contains, since each active class consumes its setup weight and contributes its setup cost. We encode these class activations directly in the DAG by introducing, for each class, a setup arc that must be traversed before any item of that class can be packed. Thus, each feasible packing pattern, including both the packed items and the classes activated in the bin, is represented by a source-to-sink path in the DAG. The associated arc-flow formulation then selects a minimum-cost collection of feasible packing patterns, while assignment constraints ensure that every item is packed exactly once.

3.1. DAG construction

To derive the arc-flow formulation for the BPPS, we construct a DAG that is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with a *source* σ and a *sink* τ. Every source-to-sink path in this DAG encodes a feasible BPPS packing pattern, whose total weight of packed items and active classes does not exceed the bin capacity. Following the arc-flow paradigm based on dynamic programming [de Lima et al., 2022], each vertex of $\mathcal{G}$ corresponds to a dynamic programming *state*. The structure of the DAG is described by two dimensions: the *stage* dimension, which represents the sequence of setup, item, and class-transition decisions, and the *load* dimension, which records the current total weight in the bin. Specifically, given a fixed ordering of the items, in the standard DAG construction for the BPP a state is identified by a pair (ℓ, t), where $\ell \in \{0, 1, \ldots, d\}$ is the accumulated load, i.e., the total item weight packed so far, and t is the stage, i.e., the position in the item ordering after the first t items have been considered. In the DAG for the BPPS, we consider an ordering of the classes and of the items within each class. The sequence of stages is then enriched by adding setup stages and class-transition stages. Furthermore, in the BPPS DAG, the accumulated load accounts for both the packed item weights

and the active classes. To account for class setups, we add one state component: two states with the same stage and load may lead to different subsequent decisions depending on whether the current class setup has already been activated and whether at least one item of that class has already been packed. We therefore extend each state with an *active flag* $e \in \{-1, 0, 1\}$, so that every vertex $v \in \mathcal{V}$ is characterized by a triple (ℓ, t, e). The active flag e reflects the status of the current class along the path: $e = -1$ marks an entry/exit vertex, $e = 0$ marks a vertex at which the setup weight of the current class has been reserved but no item of that class has yet been packed, and $e = 1$ marks a vertex at which at least one item of the current class has been packed.

All vertices sharing the same stage index t form *layer* t. These layers are of two types. For each class $c \in \mathcal{C}$, the *class layer* of c consists of $|\mathcal{I}_c| + 1$ consecutive layers: one *setup layer*, in which the setup weight of class c is reserved, followed by $|\mathcal{I}_c|$ *item layers*, one per item of class c. Separating consecutive class layers are single *entry/exit layers*. These layers contain the boundary vertices from which a class can either be entered through its setup arc or bypassed entirely. Setup and item arcs connect consecutive layers, whereas class-bypass arcs connect directly the entry and exit layers of a class. The source is $\sigma = (0, 0, -1)$. Each arc $a \in \mathcal{A}$ has a *weight* $\omega_a \in \mathbb{Z}_{\geq 0}$ and a *cost* $\lambda_a \in \mathbb{Z}_{\geq 0}$. The weight ω_a represents the load consumed by traversing arc a, while the cost λ_a represents the contribution of arc a to the objective function. In particular, setup arcs which model the activation of class $c \in \mathcal{C}$ have weight s_c and cost f_c, item arcs which model the packing of item $i \in \mathcal{I}$ have weight w_i and zero cost, bypass arcs have zero weight and zero cost, and loss arcs to the sink have zero weight and cost r. For each item $i \in \mathcal{I}$, we denote by $\mathcal{A}_{\mathcal{I}}(i) \subseteq \mathcal{A}$ the set of item arcs that pack item i. Similarly, for each class $c \in \mathcal{C}$, we denote by $\mathcal{A}_{\mathcal{C}}(c) \subseteq \mathcal{A}$ the set of setup arcs that activate class c.

The DAG is built by processing classes $c \in \mathcal{C}$ in the order induced by the input instance. Let t_{prev} denote the stage index of the current entry/exit layer, initialized to $t_{\text{prev}} = 0$. For each class c, the construction proceeds as follows.

A *bypass arc* (weight 0, cost 0) from each entry/exit vertex $(\ell, t_{\text{prev}}, -1)$ to $(\ell, t_{\text{exit}}, -1)$, where $t_{\text{exit}} = t_{\text{prev}} + |\mathcal{I}_c| + 2$, encodes packing patterns in which class c is absent. For each entry/exit vertex $(\ell, t_{\text{prev}}, -1)$ with $\ell + s_c \leq d$, a *setup arc* (weight s_c, cost f_c) to $(\ell + s_c,\ t_{\text{prev}} + 1,\ 0)$ reserves the setup weight of class c and activates it in the current bin.

Items of class c are then processed in non-increasing order of weight, starting from stage $t = t_{\text{prev}} + 1$. For each item $i \in \mathcal{I}_c$ and every vertex (ℓ, t, e) at stage t, two arcs are added: a bypass arc to

$(\ell, t+1, e)$, skipping item i; and, if $\ell + w_i \leq d$, an *item arc* (weight w_i, cost 0) to $(\ell + w_i,\ t+1,\ 1)$, packing item i. After processing all items of class c, a bypass arc from each vertex $(\ell, t_{\text{exit}} - 1, 1)$ transitions to the exit layer $(\ell, t_{\text{exit}}, -1)$. Vertices with flag $e = 0$ at stage $t_{\text{exit}} - 1$—representing states in which class c was activated but none of its items was packed—are dominated and receive no outgoing arc.

After processing class c, its exit layer becomes the entry layer for the next class, and t_{prev} is updated to t_{exit}. After the last class, a *loss arc* (weight 0, cost r) connects each vertex of the final entry/exit layer with load $\ell \geq 1$ to τ. Thus, the bin-opening cost is charged exactly once for every non-empty source-to-sink path, independently of the first class activated along the path.

Finally, a backward breadth-first search from τ marks all vertices from which the sink is reachable; all unmarked vertices and their incident arcs are then removed. This step eliminates residual vertices at which a class was activated but the DAG contains no forward path reaching τ with at least one item of that class packed. As a consequence, in the resulting uncompressed DAG, every path traversing a setup arc of class c packs at least an item $i \in \mathcal{I}_c$.

Figure 4 shows the DAG obtained after the construction phase for the example instance of Figure 1.

3.2. DAG compression

The DAG, built as described in Section 3.1, is then compressed in two successive steps, corresponding to Steps 3 and 4 of Brandão and Pedroso [2016], respectively. For each compression step, we refer to the tuple of vertex attributes used to determine whether two vertices can be merged as the *merge key*. Since the BPPS DAG contains explicit layers for setup, item, and class-bypass decisions, we keep the stage component in the merge key. Thus, vertices belonging to different layers are never merged. This is a crucial difference with respect to the compression scheme of Brandão and Pedroso [2016], where vertices may be merged even if they belong to different stages, provided that they have the same source and sink labels. In our setting, merging states of different stages would disregard the role of class setup costs and could therefore weaken the resulting arc-flow formulation.

The idea behind the compression is to identify vertices that play the same role relative to the rest of a source-to-sink path. In the first step, vertices are relabeled according to the maximum weight that can still be accumulated on a path from that vertex to the sink. If two vertices are in the same layer t and have the same value of this label, then the same amount of capacity must be left for completing any continuation from them; merging them only moves unused capacity to the beginning of the

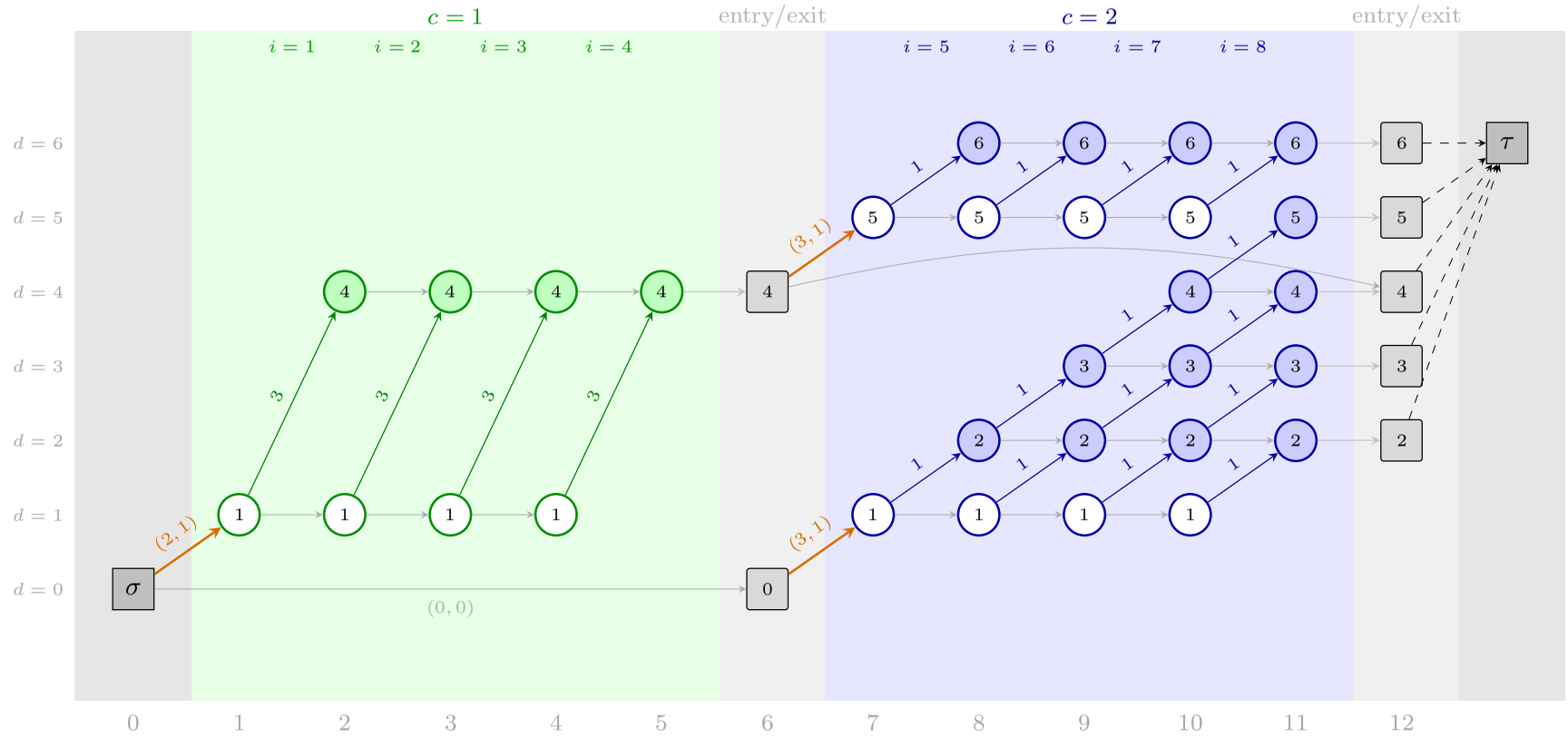


Figure 4: DAG $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ constructed for the BPPS instance of Figure 1, with bin capacity $d = 6$, $n = 8$ items partitioned into $m = 2$ classes ($\mathcal{I}_1 = \{1, 2, 3, 4\}$, $\mathcal{I}_2 = \{5, 6, 7, 8\}$), item weights $w_1 = w_2 = w_3 = w_4 = 3$ and $w_5 = w_6 = w_7 = w_8 = 1$, setup weights $s_1 = s_2 = 1$, and setup costs $f_1 = 2$, $f_2 = 3$. The DAG is shown before the compression steps of Section 3.2. Each vertex $(\ell, t, e) \in \mathcal{V}$ is placed at height ℓ on the vertical axis and at position t on the horizontal axis; stages are indicated at the bottom. Vertex colors encode the active flag e: entry/exit vertices ($e = -1$) are drawn as gray squares; interior vertices with $e = 0$ (setup weight reserved, no item packed yet) are drawn as white circles; vertices with $e = 1$ (at least one item packed) are drawn as filled circles, green for class 1 and blue for class 2. Arc colors encode the arc type. Class setup arcs are drawn in orange and labeled with the pair (λ_a, ω_a): the setup arc of class 1 has label $(2, 1)$, reflecting setup cost $f_1 = 2$ and weight $s_1 = 1$. The setup arc of class 2 has label $(3, 1)$, reflecting cost $f_2 = 3$ and weight $s_2 = 1$. Item arcs ($a \in \mathcal{A}_{\mathcal{I}}(i)$) are drawn in the color of their class and labeled with $\omega_a = w_i$: arcs corresponding to items of class 1 are green and have label 3, while arcs corresponding to items of class 2 are blue and have label 1. Bypass arcs are drawn in gray. Loss arcs, connecting the final entry/exit layer to the sink τ, are drawn as dashed black arcs. Each has zero weight and incurs the bin cost $r = 10$; this pair is not displayed as an arc label to keep the figure readable. The bypass arc from σ to the entry/exit vertex at stage $t = 6$, labeled $(0, 0)$ in gray, encodes bins in which class 1 is entirely absent. Thus, arcs leaving σ incur either zero cost or the setup cost of class 1, while the bin cost is charged only on the loss arcs entering τ.

path, as in the compression procedure of Brandão and Pedroso [2016]. The second step applies the symmetric idea from the source side. After unused capacity has been shifted toward the beginning of the paths by the first compression step, vertices are relabeled according to the maximum weight that can be accumulated from the source to the vertex. Vertices in the same layer with the same source label are indistinguishable with respect to the load that can be accumulated before reaching them, and can therefore be merged. Keeping the stage in both merge keys preserves the order in which setup and item decisions are taken.

The first compression step works as follows. For each vertex $u \in \mathcal{V}$, define the *sink label* $\phi(u, \tau)$ as the weight of the maximum weight path from u to the sink τ in $\mathcal{G}$. Notice that $d - \phi(u, \tau)$ corresponds to the maximum load that a bin can have upon reaching u in order for every suffix reachable from u to remain within capacity. Two vertices $u = (\ell, t, e)$ and $u' = (\ell', t', e')$ are then merged if and only if they share the same stage and the same sink label, i.e., $t = t'$ and $\phi(u, \tau) = \phi(u', \tau)$, resulting in a single new vertex $v = (d - \phi(u, \tau),\, t,\, \cdot)$. Feasibility is preserved under merging: any path entering the merged vertex v has total weight at most $d - \phi(u, \tau)$, and any path leaving from that vertex has total arc weight at most $\phi(u, \tau)$, so the combined weight does not exceed d.

The second compression step works as follows. On the DAG resulting from the sink-label compression step, define the *source label* $\phi(\sigma, u)$ as the weight of the maximum weight path from the source σ to u in $\mathcal{G}$. Two vertices $u = (\ell, t, e)$ and $u' = (\ell', t', e')$ are then merged if and only if they share the same stage and the same source label, i.e., $t = t'$ and $\phi(\sigma, u) = \phi(\sigma, u')$, resulting in a single new vertex $v = (\phi(\sigma, u),\, t,\, \cdot)$. Feasibility is preserved by an argument analogous to that of the sink-label compression step. The stage component in the merge key preserves the order of class and item layers after compression.

We deliberately do not include the active flag in the merge keys. Consequently, compression may introduce source-to-sink paths that traverse the setup arc of a class without traversing any of its item arcs. Such paths do not correspond to proper BPPS packing patterns, as they contain an unnecessary class activation. However, when setup weights and setup costs are nonnegative, every such path is dominated by one that bypasses the unused class, thereby consuming no more capacity and incurring no greater cost. We therefore accept these dominated paths in exchange for the greater reduction in DAG size obtained by disregarding the active flag during compression.

Figure 5 shows the DAG obtained after the compression phase for the example instance of Figure 1.

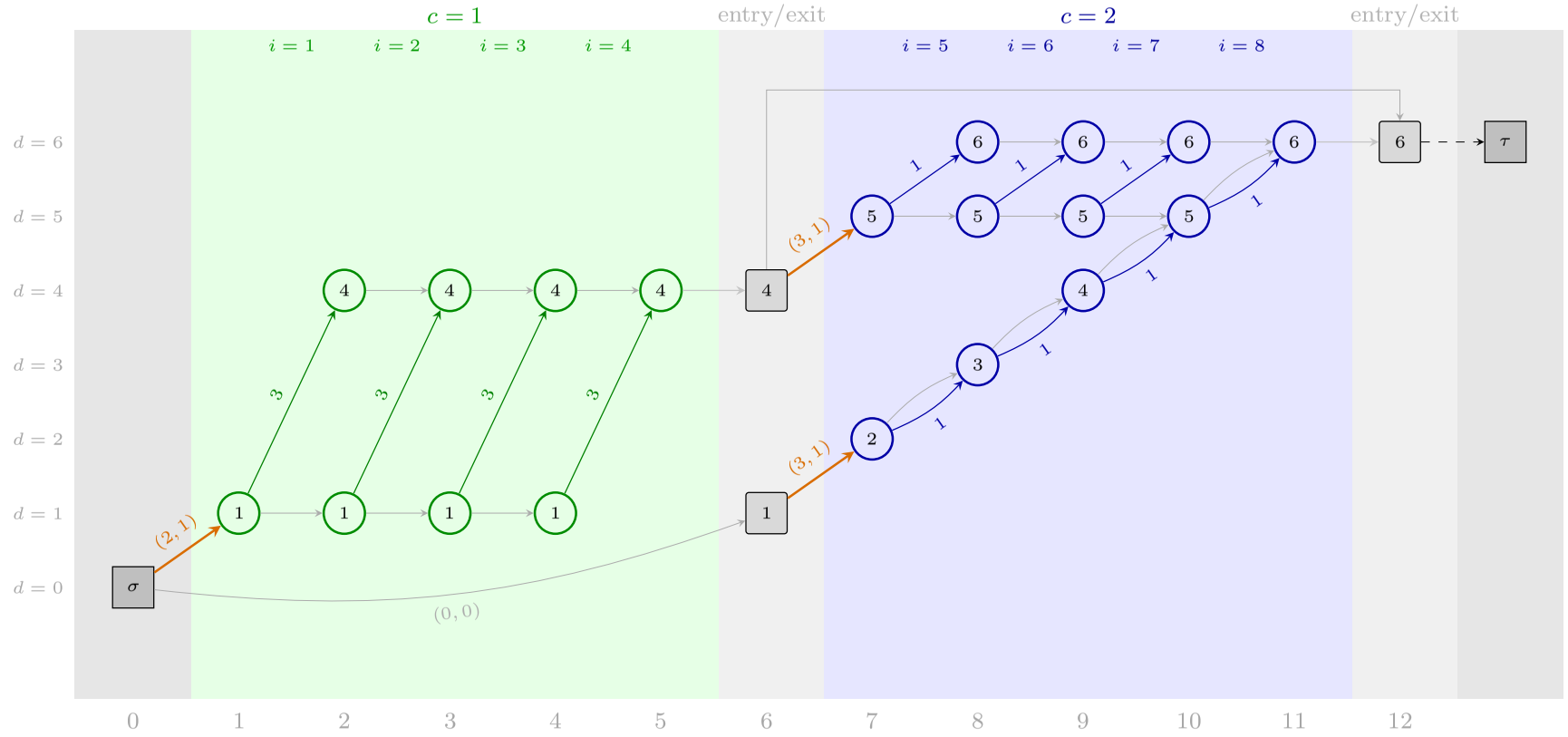


Figure 5: Compressed DAG $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ for the BPPS instance of Figure 1, obtained after applying the sink-label and source-label compression steps detailed in Section 3.2 to the DAG of Figure 4. Vertex placement follows the same convention as in Figure 4: the vertical axis reports the load ℓ and the horizontal axis reports the stage t. After compression, the active flag e is not displayed via different vertex colors, as it carries no information in the compressed DAG. Vertex and arc colors follow the same convention as in Figure 4. The effect of compression is visible in the class 2 layer ($t \in \{7, 8, \ldots, 11\}$): the 22 vertices present in the original DAG are reduced to 11 vertices. In the class 1 layer ($t \in \{1, 2, \ldots, 5\}$), the structure is unchanged, as there are no vertices at the same stage which can be merged.

3.3. The arc-flow ILP formulation

For each arc $a \in \mathcal{A}$ of the compressed DAG, let $\varphi_a \in \mathbb{Z}_{\geq 0}$ be an integer variable which represents the flow traversing arc a. The arc-flow formulation for the BPPS, which we refer to as $\mathrm{ILP}_{\mathrm{AF}}$, reads:

$$(\mathrm{ILP}_{\mathrm{AF}}) \qquad \min_{\boldsymbol{\varphi}} \quad \sum_{a \in \mathcal{A}} \lambda_a \, \varphi_a \tag{13a}$$

$$\sum_{a \in \delta^-(v)} \varphi_a - \sum_{a \in \delta^+(v)} \varphi_a \;=\; 0, \qquad v \in \mathcal{V} \setminus \{\sigma, \tau\}, \tag{13b}$$

$$\sum_{a \in \mathcal{A}_{\mathcal{I}}(i)} \varphi_a \;=\; 1, \qquad i \in \mathcal{I}, \tag{13c}$$

$$\varphi_a \in \mathbb{Z}_{\geq 0}, \qquad a \in \mathcal{A}, \tag{13d}$$

where $\delta^-(v)$ and $\delta^+(v)$ denote the sets of arcs entering and leaving vertex v, respectively. The objective function (13a) coincides with the total cost, which comprises the bin-opening costs and the class setup costs. The *flow conservation constraints* (13b) ensure that each unit of flow from σ to τ represents one bin, and the total outflow from σ equals the number of bins used. The *assignment constraints* (13c) require each item $i \in \mathcal{I}$ to be packed exactly once; these constraints are defined over the item arcs $\mathcal{A}_{\mathcal{I}}(i)$—which, by construction, are the only arcs encoding the packing of item i. Since $\mathcal{G}$ is acyclic, any feasible solution $\boldsymbol{\varphi}$ to $\mathrm{ILP}_{\mathrm{AF}}$ can be decomposed into $\sum_{a \in \delta^+(\sigma)} \varphi_a$ source-to-sink paths. Each path represents one used bin and determines a feasible packing pattern $S \subseteq \mathcal{I}$ consisting of the items whose item arcs are traversed by the path. By the assignment constraints (13c), these packing patterns are pairwise disjoint and cover $\mathcal{I}$; hence, they define a feasible partition $\mathcal{S}$ in the sense introduced in Section 1. Overall, $\mathrm{ILP}_{\mathrm{AF}}$ involves $|\mathcal{A}|$ integer variables and $|\mathcal{V}| - 2 + |\mathcal{I}|$ linear constraints. The size of the DAG is pseudo-polynomial in the input data. Indeed, before compression, the DAG has one layer for each item and a linear number of additional layers for setup and class-transition decisions. Within each layer, the cumulative load can take at most $d + 1$ values, and the active flag has only a constant number of values. Moreover, each state generates only a constant number of outgoing arcs. Thus, the uncompressed DAG has a size of order $O((n + m)d)$, and the compression steps can only reduce this size.

3.4. Strength of the LP relaxation of $\mathrm{ILP}_{\mathrm{AF}}$

After describing the arc-flow formulation, we now compare the strength of the LP relaxation of the arc-flow formulation with that of the natural formulation introduced in Section 2. The next

proposition formalizes that this relaxation is at least as strong as the compact one. The proof shows that every feasible solution of $\mathrm{LP_{AF}}$ has an objective function value at least equal to the optimal objective function value of $\mathrm{LP_N}$ (see (5)); taking the minimum over all feasible solutions of $\mathrm{LP_{AF}}$ then yields the result. Related dominance results have also been established for the BPP in the literature by exploiting the connection between arc-flow formulations and the Dantzig–Wolfe reformulation of the underlying compact model; see, e.g., Valério de Carvalho [1999]. Let $\zeta(\mathrm{LP_{AF}})$ denote the optimal objective function value of the LP relaxation of $\mathrm{ILP_{AF}}$.

Proposition 11. *The LP relaxation of the arc-flow formulation* $\mathrm{ILP_{AF}}$ *is at least as strong as the LP relaxation of the natural compact formulation* $\mathrm{ILP_N}$*, i.e.,*

$$\zeta(\mathrm{LP_{AF}}) \geq \zeta(\mathrm{LP_N}).$$

Proof. Consider an arbitrary feasible solution $\boldsymbol{\varphi}$ of $\mathrm{LP_{AF}}$. Since $\mathcal{G}$ is acyclic, any feasible flow on $\mathcal{G}$ can be decomposed into a collection $\mathcal{Q}$ of source-to-sink paths with nonnegative path-flow values ξ_P, $P \in \mathcal{Q}$. Each path P is identified with the set of arcs it contains, so that $P \subseteq \mathcal{A}$. Hence,

$$\varphi_a = \sum_{P \in \mathcal{Q}:\, a \in P} \xi_P, \qquad \sum_{P \in \mathcal{Q}} \xi_P = \sum_{a \in \delta^-(\tau)} \varphi_a.$$

For every $P \in \mathcal{Q}$, let $\mathcal{I}(P)$ and $\mathcal{C}(P)$ be, respectively, the items and classes represented by its item and setup arcs. For each path $P \in \mathcal{Q}$, we have that $\sum_{i \in \mathcal{I}(P)} w_i + \sum_{c \in \mathcal{C}(P)} s_c \leq d$ by construction. Multiplying this inequality by ξ_P, summing over $\mathcal{Q}$, and using the path decomposition arc by arc gives

$$\sum_{i \in \mathcal{I}} w_i \sum_{a \in \mathcal{A}_{\mathcal{I}}(i)} \varphi_a + \sum_{c \in \mathcal{C}} s_c \sum_{a \in \mathcal{A}_{\mathcal{C}}(c)} \varphi_a \leq d \sum_{a \in \delta^-(\tau)} \varphi_a. \tag{14}$$

The assignment constraints give $\sum_{a \in \mathcal{A}_{\mathcal{I}}(i)} \varphi_a = 1$ for every $i \in \mathcal{I}$. Moreover, for each $c \in \mathcal{C}$, the assignment of any item $i \in \mathcal{I}_c$ imply $\sum_{a \in \mathcal{A}_{\mathcal{C}}(c)} \varphi_a \geq 1$. Thus, by (14) and $s_c \geq 0$ for all $c \in \mathcal{C}$, we have

$$\sum_{i \in \mathcal{I}} w_i + \sum_{c \in \mathcal{C}} s_c \leq d \sum_{a \in \delta^-(\tau)} \varphi_a.$$

Only loss arcs entering τ and setup arcs have nonzero costs; therefore, since $f_c \geq 0$ for all $c \in \mathcal{C}$, we

have

$$\sum_{a\in\mathcal{A}} \lambda_a \varphi_a = r \sum_{a\in\delta^-(\tau)} \varphi_a + \sum_{c\in\mathcal{C}} f_c \sum_{a\in\mathcal{A}_\mathcal{C}(c)} \varphi_a \geq \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c = \zeta(\mathrm{LP_N}),$$

where the last equality follows from Proposition 1. Taking the minimum over the feasible solutions of $\mathrm{LP_{AF}}$ proves the result. □

The dominance can be strict, as illustrated by the two instances in the example of Figure 1. Specifically, in the instance of part (a), $\zeta(\mathrm{LP_N}) = 35$, while $\zeta(\mathrm{LP_{AF}}) = 60$; similarly, in the instance of part (b), $\zeta(\mathrm{LP_N}) = 8$, while $\zeta(\mathrm{LP_{AF}}) = 16$. This is also reflected in the computational results: at the price of a larger number of variables and constraints, the arc-flow formulation provides a consistently stronger LP relaxation than the natural compact formulation across the tested instances; see Section 4.3.

3.5. Valid inequalities

The valid inequalities introduced for the natural formulation can be transferred to $\mathrm{ILP_{AF}}$. Their validity follows from the same arguments used in Sections 2.2 and 2.4, since the corresponding arc-flow expressions count, respectively, the number of activations of each class and the number of used bins. Specifically, the MCIs (6) become

$$\sum_{a\in\mathcal{A}_\mathcal{C}(c)} \varphi_a \;\geq\; \gamma_c, \qquad c \in \mathcal{C}, \tag{15}$$

while the MBI translates to a lower bound on the total outflow from the source σ:

$$\sum_{a\in\delta^+(\sigma)} \varphi_a \;\geq\; \left\lceil \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c}{d} \right\rceil. \tag{16}$$

We denote by $\mathrm{ILP^{\dagger}_{AF}}$ the formulation obtained by adding the MCIs (15) to $\mathrm{ILP_{AF}}$, and by $\mathrm{ILP^{\star}_{AF}}$ the formulation obtained by adding both the MCIs (15) and the MBI (16) to $\mathrm{ILP_{AF}}$. Their LP relaxations are denoted by $\mathrm{LP^{\dagger}_{AF}}$ and $\mathrm{LP^{\star}_{AF}}$, respectively, and their optimal objective function values by $\zeta(\mathrm{LP^{\dagger}_{AF}})$ and $\zeta(\mathrm{LP^{\star}_{AF}})$.

4. Computational experiments

In this section, we present the results of the computational experiments carried out to evaluate the performance of the ILP formulations for the BPPS problem introduced in Sections 2 and 3, along with the proposed enhancements: the MCIs (see (6) and (15)), the MBI (see (9) and (16)), 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 models (2) and (13) by means of a general-purpose ILP solver. When optimality cannot be certified within the imposed time limit, we assess solution quality using the optimality gap. Our experiments pursue three main goals: (i) assessing the computational effectiveness of the proposed ILP formulations and their variants by identifying the largest instance sizes (and their structural features) that can be solved to optimality within a given time limit and determining the features that most affect problem difficulty; (ii) evaluating the relative performance of the different variants of the natural formulation ILP_N and the arc-flow formulation ILP_AF against each other, to identify the families of instances on which each model performs best (see Section 4.2); and (iii) assessing the strength of the LP relaxations of the two formulation families and its relation to their computational performance (see Section 4.3).

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 formulations. 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.

4.1. Library of benchmark BPPS instances

In this section, we introduce the library of benchmark instances specifically designed for the BPPS, comprising both randomly generated instances and real-world instances from a vehicle-routing application. 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 formulations and an assessment of how different instance features influence computational performance.

We propose a set of randomly generated instances, obtained by extending the approach proposed in Schwerin and Wäscher [1997] for the classical BPP. We therefore adopted a methodology consistent

with the literature on bin packing problems while incorporating item classes and their associated setup weights and costs. Each BPPS instance 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 to ensure feasibility. In line with Schwerin and Wäscher [1997], we introduce parameters $\upsilon_1, \upsilon_2, \eta_1, \eta_2 \in [0, 1]$, with $\upsilon_1 < \upsilon_2$ and $\eta_1 < \eta_2$, 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 $[\eta_1\, d,\ \eta_2\, d] \cap \mathbb{Z}$.

To ensure instance feasibility, we enforce the condition $\upsilon_2 + \eta_2 \leq 1$. Each item is assigned to one of the m available classes, with the assignment chosen uniformly at random. Every instance is generated either with or without setup 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 minimizes the number of bins used, since no setup costs are incurred. We generated BPPS instances for all combinations of the following instance-generator parameters: the number of items $n \in \{25,\ 50,\ 100,\ 200\}$, 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), four item size categories—*small* ($\upsilon_1 = 0.01$, $\upsilon_2 = 0.10$), *medium* ($\upsilon_1 = 0.10$, $\upsilon_2 = 0.20$), *large* ($\upsilon_1 = 0.20$, $\upsilon_2 = 0.30$) and *mixed* ($\upsilon_1 = 0.01$, $\upsilon_2 = 0.30$)—and three setup weight categories—*small* ($\eta_1 = 0.01$, $\eta_2 = 0.10$), *large* ($\eta_1 = 0.10$, $\eta_2 = 0.20$) and *mixed* ($\eta_1 = 0.01$, $\eta_2 = 0.20$). One instance was generated for each combination of these parameters, resulting in a total of 576 random instances.

We also consider a set of 36 real-world instances arising from a vehicle-routing application in the distribution of fresh and frozen grocery products, based on data provided by a major international grocery retailer. The operational setting involves transferring goods between distribution terminals using two types of vehicles:

I) a six-axle non-refrigerated multi-trailer vehicle with dimensions $(7.5\,\text{m} + 7.5\,\text{m}) \times 2.44\,\text{m} \times 3\,\text{m}$, capable of carrying up to 36 Euro pallets.

II) a six-axle refrigerated tractor–semitrailer whose internal loading compartment measures $13.6\,\text{m} \times 2.44\,\text{m} \times 2.6\,\text{m}$ (length × width × height) and accommodates up to 32 Euro pallets in a single floor layer. When both product types are carried simultaneously, a movable bulkhead separates the two temperature zones. Since the bulkhead can be repositioned between any

two consecutive pallet rows, the vehicle can flexibly accommodate any mix of fresh and frozen products, subject to the overall pallet capacity.

Pallets cannot be stacked and are arranged in rows of two. Type I vehicles carry fresh products only, while Type II vehicles may carry both fresh and frozen products simultaneously. Using a Type II vehicle incurs a higher cost than using a Type I vehicle. In this scenario, the daily set of customer orders is expressed in Euro-pallet units and defines the item set. Each order is classified as either *fresh* or *frozen*, yielding two item classes ($m = 2$): items in class $c = 1$ (fresh) can be loaded into either vehicle type. Because fresh orders impose no refrigeration requirement, they do not consume extra capacity and incur no additional cost when placed in a bin. Hence, the setup weight and setup cost for class $c = 1$ are both zero ($s_1 = 0$, $f_1 = 0$). On the other hand, items from class $c = 2$ (frozen) can be loaded only into a Type II vehicle (a refrigerated bin). Hence, when at least one frozen order is assigned to a vehicle, the effective pallet capacity decreases by 4 (from 36 to 32, matching the Type II specification), and an additional cost $\alpha\, r$ is charged (we consider instances with $\alpha \in \{0.1, 0.3\}$. Thus $s_2 = 4$ and $f_2 = \alpha r$. The bin capacity is set to $d = 36$ (the Type I capacity) and the bin cost to $r = 100$. Figure 6 provides an illustration of the two vehicle types.

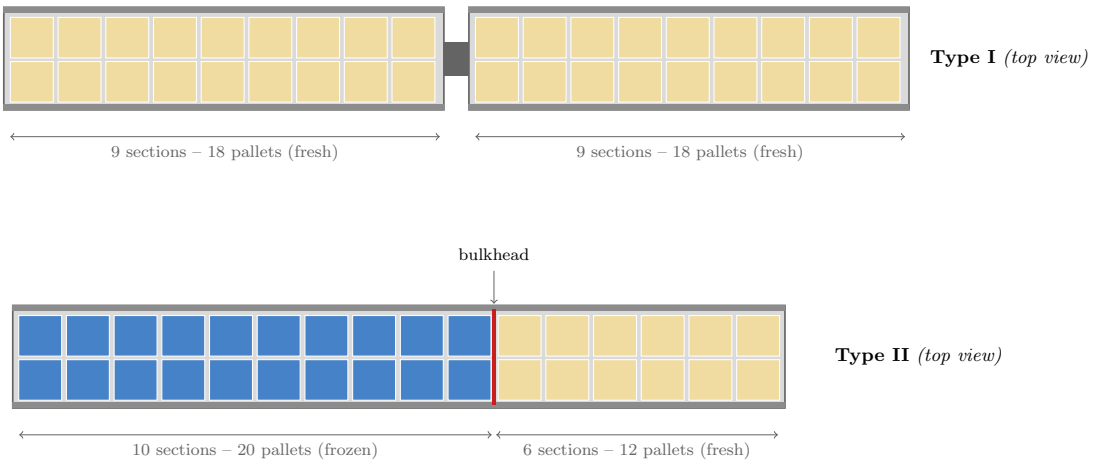


Figure 6: Schematic representation of the two vehicle types used in the real-world BPPS instances.

The real-world instances are organized into three groups of 12 instances: `rwA`, `rwB`, `rwC`. The `rwA` and `rwB` instances share the same item-class partition (i.e., the same number of fresh and frozen orders for every instance). They differ in the minimum pallet demand per customer order: specifically, every fresh and frozen order in `rwA` contains at least 12 pallets, whereas in `rwB` the minimum is 14 pallets for fresh orders and 10 pallets for frozen orders. In `rwC`, some pallet requirements are

fractional. To express all BPPS input data as integers, pallet quantities are multiplied by 10. Thus, $d = 360$ represents a bin capacity of 36 pallets, $s_2 = 40$ represents a setup weight of 4 pallets, and the minimum item weight of 112 represents an order of 11.2 pallets.

The complete benchmark set and its documentation, along with the random instance generator toolkit, are publicly available at the following GitHub repository: `https://github.com/FabioCiccarelli/BPPS`. The repository also contains the implementation of the mathematical formulations introduced in this paper, as well as the algorithms for constructing and compressing the arc-flow DAG described in Section 3. We hope that the resources made available through this repository will foster further research on the BPPS by providing a common reference framework for developing and comparing novel exact and heuristic algorithms, mathematical formulations, computational studies, and polyhedral analyses.

4.2. Computational performance of the ILP formulations

In this section, we compare the computational performance of the following seven variants of the proposed ILP formulations (2) and (13) for the BPPS. The base variant of the natural formulation, denoted by ILP_N, is formulation (2) in which the upper bound k on the number of bins in any optimal BPPS solution is set to n, namely the number of items. The first variant of the natural formulation, denoted by $\text{ILP}_\text{N}^\dagger$, extends ILP_N by including the MCIs (6). The second variant, denoted by $\text{ILP}_\text{N}^\ddagger$, extends ILP_N by including the MCIs (6) and the MBI (9). The third variant of the natural formulation, denoted by $\text{ILP}_\text{N}^\star$, extends $\text{ILP}_\text{N}^\ddagger$ by including the upper bound $\hat{k}$ 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 $\hat{k}$ were obtained using classical heuristic algorithms for the BPP, specifically the Next Fit (NF), First Fit (FF), and Best Fit (BF) algorithms (see Johnson et al. [1974]), each executed under 50 random permutations of the item order (including the non-increasing order of weights). The base variant of the arc-flow formulation is denoted by ILP_AF and corresponds to Model (13). The first variant of the arc-flow formulation, denoted by $\text{ILP}_\text{AF}^\dagger$, extends ILP_AF by including the MCIs (15). The second variant of the arc-flow formulation, denoted by $\text{ILP}_\text{AF}^\star$, extends ILP_AF by including the MCIs (15) and the MBI (16). In all seven variants, the solver is provided with a warm-start feasible solution constructed as follows. For each class $c \in \mathcal{C}$, the items of class c are packed into dedicated bins using the heuristic solution computed to derive $\overline{\beta}_c$. The resulting solution, which features single-class bins only, is passed to the solver as an initial solution.

Concerning the arc-flow formulation, it is interesting to assess the effectiveness of the compression

phase described in Section 3.2 in reducing the size of the arc-flow DAGs. Across the 576 random instances, it removes a median of 71.0% of the vertices and 67.5% of the arcs. Although the extent of the reduction varies considerably across instances, it remains substantial in all cases, ranging from 43.6% to 97.8% for vertices and from 42.1% to 96.9% for arcs.

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 solved using *Gurobi 13.0.2*, 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 seven 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. For the three arc-flow variants, the table additionally reports, for each instance group, the number of instances for which no valid dual bound could be computed within the time limit (column "n.c."); these instances are excluded from the average gap computation for the corresponding variant. The random instances are grouped according to the six instance-generator parameters described in the previous section on the benchmark instance library, while the real-world instances are grouped according to the subset they belong to (`rwA`, `rwB` or `rwC`). For each group, the table also reports the number of instances in that group (column "inst").

The results in Table 1 reveal distinct patterns depending on the instance characteristics, which we discuss separately for the natural and arc-flow formulation families before comparing them directly. Within the natural formulation family, performance differences become more pronounced as the instance size or complexity increases. For small instances ($n = 25$), all four variants perform similarly. However, as n grows, ILP_N rapidly loses effectiveness, with a 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; the further inclusion of the MBI in $\text{ILP}_\text{N}^\ddagger$ allows a larger number of instances to be solved to optimality. The combination of these valid inequalities with the tighter upper bound on the number of bins in $\text{ILP}_\text{N}^\star$ provides the best results among the natural formulation variants for most of the instance groups, both in terms of optimal solutions and reduced gap. A similar trend holds when varying the number of classes m, the bin capacity d, the setup cost structure, and the setup and item weight distributions, with $\text{ILP}_\text{N}^\star$ matching or outperforming $\text{ILP}_\text{N}^\ddagger$ across most

of the groups. Within the arc-flow family, $\text{ILP}^{\star}_{\text{AF}}$ is consistently the strongest variant. A notable limitation, however, is the significant number of instances for which no valid dual bound could be computed, particularly for larger values of n (up to 66 instances out of 144 for $n = 200$) and larger bin capacities (up to 71 out of 192 for $d = 10,000$).

A further comparison can be done between the two formulations introduced in this article, namely the natural model and the arc-flow model. The item weight distribution emerges as the dominant factor: for instances with large item weights, the arc-flow variants are substantially more effective, with $\text{ILP}^{\star}_{\text{AF}}$ solving 132 out of 144 instances compared to 34 for $\text{ILP}^{\star}_{\text{N}}$, and with an average optimality gap of 1.3% against 6.7%. Conversely, for small and medium item weights, the natural formulation variants are definitely superior, and the arc-flow formulation suffers from a higher rate of instances marked as "n.c.". This difference explains the behavior on the real-world instances, which feature large item weights: the arc-flow variant $\text{ILP}^{\star}_{\text{AF}}$ solves all 36 instances to optimality, while the best natural formulation variant solves only 8.

Overall, across the 576 random instances, 503 (87.3%) are solved to proven optimality by at least one of the seven configurations, and all 36 real-world instances are solved as well. The remaining 73 random instances (12.7%) are not solved by any configuration within the time limit. The minimum optimality gap on these instances ranges from 0.27% to 6.67% (average 2.85%, median 2.78%), so the solver remains close to optimality even on these harder instances. Difficulty is influenced almost entirely by the values of n and m: no instance with $n \leq 50$ remains unsolved, while the unsolved rate reaches 44.4% for $n = 200$, $m = 10$ and 36.1% for $n = 100$, $m = 10$ and bin-setup costs, reflecting the higher complexity induced by many classes with many items each. Two secondary factors further contribute to instance difficulty: adding the bin-setup costs raises the unsolved rate from 9.7% to 15.6%, and instances with medium or mixed item-weight ranges are harder (20.8% and 20.1% unsolved, respectively) than small or large ones (2.1% and 7.6%). No single formulation consistently provides better bounds on the open instances: $\text{ILP}^{\dagger}_{\text{N}}$ yields the best optimality gap in 23 of the 73 cases and $\text{ILP}^{\star}_{\text{AF}}$ in 18, confirming the complementary behavior of the natural and arc-flow models discussed above.

Figure 7 plots the number of random instances solved as a function of time for the seven ILP formulation variants. The x-axis represents the computing time (in seconds, on a logarithmic scale), and the y-axis reports the cumulative number of instances solved to optimality within that time. Among the natural formulation variants, the ranking is consistent throughout almost the entire time

Table 1: Performance comparison of the seven ILP formulation variants for the BPPS, reporting the number of instances solved to optimality (opt), the average optimality gap over unsolved instances (gap) and, for the arc-flow variants, the number of instances for which no valid dual bound could be computed within the time limit (n.c.). The instances are grouped by instance generator parameters. Time limit: 1800 seconds.

			Natural								Arc-flow								
			ILP_N		$ILP_N^\dagger$		$ILP_N^\ddagger$		$ILP_N^\star$		ILP_{AF}			$ILP_{AF}^\dagger$			$ILP_{AF}^\star$		
		inst	opt	gap	opt	gap	opt	gap	opt	gap	opt	gap	n.c.	opt	gap	n.c.	opt	gap	n.c.
`item number`	$n = 25$	144	137	10.7	136	10.4	137	11.1	137	11.1	**144**	-	0	143	3.9	0	**144**	-	0
	$n = 50$	144	83	8.3	99	7.2	107	6.8	107	6.7	108	4.4	9	119	4.5	9	**128**	7.1	9
	$n = 100$	144	43	11.2	65	4.8	77	4.9	**79**	5.0	52	6.3	24	57	3.9	36	69	3.7	36
	$n = 200$	144	18	16.2	54	4.1	**64**	4.4	63	4.5	27	5.5	45	25	4.6	66	38	4.8	64
`class number`	$m = 5$	288	150	11.8	200	5.2	**214**	5.3	**214**	5.5	161	5.4	40	170	4.0	57	200	4.2	56
	$m = 10$	288	131	13.5	154	5.2	171	5.3	172	5.2	170	6.0	38	174	4.5	54	**179**	4.6	53
`bin capacity`	$d = 200$	192	93	13.1	125	5.0	135	4.9	134	5.1	128	6.1	0	131	4.9	0	**148**	5.0	0
	$d = 1{,}000$	192	95	12.9	120	5.3	127	5.4	**129**	5.4	106	5.9	18	111	4.0	40	122	4.3	38
	$d = 10{,}000$	192	93	12.1	109	5.4	**123**	5.5	**123**	5.5	97	4.4	60	102	2.8	71	109	2.7	71
`bin-setup costs`	no	288	150	8.3	184	6.2	190	6.1	189	6.1	**208**	6.1	42	207	5.8	56	200	6.3	55
	yes	288	131	16.6	170	4.3	195	4.4	**197**	4.4	123	5.5	36	137	3.9	55	179	3.4	54
`item weights`	small	144	124	11.2	139	6.5	140	6.0	**141**	5.2	70	11.7	50	68	8.8	55	72	10.3	55
	medium	144	63	11.4	90	3.5	102	3.4	**105**	3.4	78	5.6	8	82	4.1	22	93	3.9	21
	large	144	30	14.3	30	6.8	35	6.5	34	6.7	110	1.4	0	118	1.5	0	**132**	1.3	0
	mixed	144	64	12.1	95	3.4	**108**	3.6	106	3.3	73	5.7	20	76	3.9	34	82	2.9	33
`setup weights`	small	192	98	11.4	123	5.3	**129**	5.4	**129**	5.6	104	5.6	32	109	4.9	44	118	4.3	43
	large	192	87	14.0	114	5.2	124	5.1	127	5.1	121	5.7	17	123	3.8	29	**134**	4.5	31
	mixed	192	96	12.6	117	5.2	**132**	5.4	130	5.3	106	5.7	29	112	4.1	38	127	4.6	35
Total/Average		576	281	12.7	354	5.2	385	5.3	**386**	5.3	331	5.7	78	344	4.3	111	379	4.5	109
`real-world`	`rwA`	12	2	10.2	2	5.7	1	7.5	4	6.5	6	0.3	0	11	0.4	0	**12**	-	0
	`rwB`	12	2	7.1	1	9.4	1	5.1	4	1.1	**12**	-	0	**12**	-	0	**12**	-	0
	`rwC`	12	0	7.0	1	7.5	1	7.1	0	7.7	11	0.2	0	**12**	-	0	**12**	-	0
Total/Average		36	4	8.0	4	7.6	3	6.6	8	5.5	29	0.3	0	35	0.4	0	**36**	-	0

horizon: $ILP_N^\star$ dominates, followed by $ILP_N^\ddagger$, $ILP_N^\dagger$, and ILP_N. The gap between $ILP_N^\star$ and ILP_N is substantial and emerges within the first few seconds, showing that the proposed strengthening techniques are also effective on the easier instances. Within the arc-flow family, $ILP_{AF}^\star$ similarly dominates $ILP_{AF}^\dagger$ and ILP_{AF} throughout the time horizon, although the separation among the three variants is less pronounced than within the natural family. Comparing the two formulation families on the random instances, $ILP_N^\star$ and $ILP_N^\ddagger$ achieve the best performance in the early phase, below approximately one second, solving more instances than any arc-flow variant. Specifically, within the first 0.1 seconds, $ILP_N^\star$ solves around 150 instances, compared with approximately 50 for $ILP_{AF}^\star$. As time increases, the arc-flow variants narrow the gap, and $ILP_{AF}^\star$ ultimately solves almost as many instances as $ILP_N^\star$ by the end of the time horizon. These profiles therefore show that, on the

random benchmark, the strongest natural formulation variants are more effective on instances solved within short computing times, whereas the performance of $\text{ILP}^{\star}_{\text{N}}$ and $\text{ILP}^{\star}_{\text{AF}}$ becomes comparable over longer time horizons. This conclusion does not extend to all instance groups: as shown in Table 1, the relative performance of the two formulation families depends mainly on the item weight distribution, with the arc-flow variants outperforming the natural ones on instances with large item weights and on the real-world testbed.

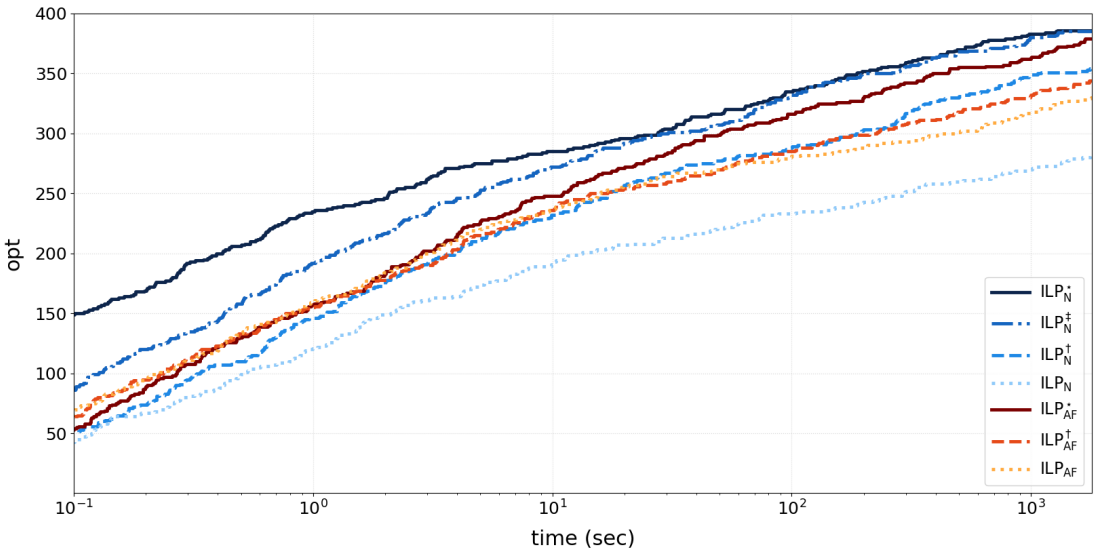


Figure 7: Number of random instances solved as a function of time for the different variants of the ILP formulations for the BPPS. Time limit: 1800 seconds. The horizontal axis is on a logarithmic scale.

4.3. Strength of the LP relaxations and features of the optimal solutions

We conclude the computational analysis with an assessment of the LP relaxation strength of the seven variants, as tighter LP bounds generally correlate with faster branch-and-bound convergence and higher rates of optimally solved instances. Moreover, we briefly discuss the main structural features of the optimal BPPS solutions. A detailed breakdown is reported in Section E.3 of the electronic companion. The analysis is conducted on 427 randomly selected instances for which an optimal solution is known, and an optimal LP solution can be computed for all arc-flow formulation variants within a 30-minute time limit. The average integrality gap, defined as the percentage difference between the LP relaxation value and the known optimal integer objective, taken with respect to the optimum, confirms the hierarchy observed in the computational experiments. Among the natural formulation variants, LP_{N} exhibits the largest average gap (17.8%), which is substantially reduced by the addition of the MCIs in $\text{LP}^{\dagger}_{\text{N}}$ (8.3%) and by the inclusion of the MBI in $\text{LP}^{\star}_{\text{N}}$ (3.0%). The arc-flow variants have tighter LP relaxations consistently, confirming the theoretical dominance

stated in Proposition 11: $\mathrm{LP}_{\mathrm{AF}}$ already achieves an average gap of 6.6%, comparable to $\mathrm{LP}_{\mathrm{N}}^{\dagger}$, while $\mathrm{LP}_{\mathrm{AF}}^{\star}$ reaches 1.6%, the lowest among all variants. The contrast between the two families is most striking on the 36 real-world instances, where the natural formulation variants exhibit gaps in the range 9–23% even for $\mathrm{LP}_{\mathrm{N}}^{\star}$, whereas all three arc-flow variants achieve an average integrality gap below 1%; this is consistent with the dominance of the arc-flow formulation on the real-world instances, observed in Table 1. Nevertheless, a stronger LP relaxation does not automatically translate into superior computational performance. As shown in Table 1, the arc-flow variants are outperformed by the natural formulation on several instance groups. This apparent discrepancy is explained by the size of the arc-flow DAG, which can grow very large depending on the instance structure, making the LP at the root node itself computationally expensive to solve and, in the most extreme cases, preventing the computation of any valid dual bound within the time limit. The LP strength advantage of the arc-flow formulation is therefore offset, in such instances, by the excessive computational overhead caused by the DAG size.

Beyond the comparison of the LP relaxations, the statistics reported in Section E.3 of the electronic companion also provide some insight into the structure of optimal BPPS solutions. On the random instances, optimal bins are generally highly utilized, with an average fill rate of 92.4%, and contain on average 5.8 items belonging to 1.6 active classes. The real-world solutions exhibit an even more homogeneous class composition, with exactly one active class and 1.7 items per bin on average, while maintaining an average fill rate of 87.8%.

5. 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. We 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 Inequality* (MBI), which further improved the quality of the LP relaxation. Beyond strengthening the natural formulation, we developed an arc-flow formulation for the BPPS, based

on a tailored graph construction and compression procedure that encodes class activation, setup weights, and setup costs directly in the graph. We proved that the LP relaxation of this formulation dominates that of the natural formulation, and we extended the MCIs and the MBI to the arc-flow model as well. To support computational evaluation, we developed a publicly available benchmark library of 576 randomly generated BPPS instances, designed to capture a broad spectrum of instance characteristics, including variations in item weights, class distributions, setup costs, and setup weights, complemented by 36 real-world instances arising from a vehicle-routing application in the distribution of fresh and frozen grocery products. Using this library, we conducted an extensive computational campaign to assess the performance of both formulation families and the enhancements proposed for each. Our experiments show that the proposed enhancements substantially improve computational performance within both formulation families, increasing the number of instances solved to proven optimality and reducing the optimality gaps on unsolved instances. Comparing the two formulations, the item weight distribution emerges as the dominant factor: the natural formulation is generally more effective on instances with small or medium item weights, while the arc-flow formulation dominates on instances with large item weights and on the real-world testbed, where it solves all instances to optimality and consistently yields tighter LP relaxations. This complementary behavior indicates that neither formulation is consistently superior, and that the instance structure should guide the choice between them.

Future research directions include investigating the polyhedral structure of the proposed formulations, 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. The formulations and the benchmark library introduced in this paper provide a natural starting point for this line of work. Another promising avenue is to develop polynomial-time approximation algorithms with provable performance guarantees.

Acknowledgments

The authors thank the anonymous reviewers and the editor for their valuable comments and thorough review of the paper.

References

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

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.

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.

M. M. Baldi, D. Manerba, G. Perboli, and R. Tadei. A generalized bin packing problem for parcel delivery in last-mile logistics. *European Journal of Operational Research*, 274(3):990–999, 2019.

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.

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.

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

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.

F. Brandão and J. P. Pedroso. Bin packing and related problems: General arc-flow formulation with graph compression. *Comput. Oper. Res.*, 69:56–67, 2016.

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.

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

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

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

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

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.

T. G. Crainic, F. Djeumou Fomeni, and W. Rei. Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning. *Computers & Operations Research*, 132:105308, 2021.

V. L. de Lima, C. Alves, F. Clautiaux, M. Iori, and J. M. Valério de Carvalho. Arc flow formulations based on dynamic programming: Theoretical foundations and applications. *European Journal of Operational Research*, 296(1):3–21, 2022.

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.

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

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.

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, 2016a.

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, 2016b.

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.

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

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

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.

E. Lin. A bibliographical survey on some well-known non-standard knapsack problems. *INFOR: Information Systems and Operational Research*, 36(4):274–317, 1998.

S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990.

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.

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.

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

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.

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

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. *OR Spectrum*, 37(2):297–330, 2015.

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

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

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.

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

J. M. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86(0):629–659, 1999.

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.

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.

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

E.1. Notation summary

In Table E.1, we summarize the notation used throughout the paper. The first three blocks of the table list, respectively, the data of the classical BPP, the additional data specific to the BPPS, and the auxiliary parameters and valid bounds used in our article. The fourth block describes the natural ILP formulation ILP_N and its enhancements ($\text{ILP}_\text{N}^\dagger$, $\text{ILP}_\text{N}^\ddagger$, and $\text{ILP}_\text{N}^\star$). The fifth and sixth blocks introduce, respectively, the notation for the arc-flow DAG $\mathcal{G}$ and the arc-flow formulation ILP_AF and its enhancements ($\text{ILP}_\text{AF}^\dagger$ and $\text{ILP}_\text{AF}^\star$). The last block reports the optimal objective function values of the LP relaxations of all formulations, which provide lower bounds on OPT.

Table E.1: Summary of notation

BPP givens	**Description**
$d \in \mathbb{Z}_{>0}$	Bin capacity
$n \in \mathbb{Z}_{>0}$	Number of items
$\mathcal{I} = \{1, 2, \ldots, n\}$	Set of items
$w_i \in \mathbb{Z}_{>0}$	Weight of item $i \in \mathcal{I}$
$\beta \in \mathbb{Z}_{>0}$	Optimal objective function value for the BPP
BPPS givens	**Description**
$m \in \mathbb{Z}_{\geq 1}$	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
$\mathcal{S} \subseteq 2^\mathcal{I}$	Feasible BPPS partition into packing patterns
$S \in \mathcal{S}$	A single packing pattern (bin) belonging to a feasible BPPS partition $\mathcal{S}$
$\mathcal{C}(S) = \{c \in \mathcal{C} : S \cap \mathcal{I}_c \neq \emptyset\}$	Set of classes active in a subset of items $S \subseteq \mathcal{I}$
$UB(\mathcal{S}) \in \mathbb{Z}_{>0}$	Objective function value of a feasible BPPS partition $\mathcal{S}$
$OPT \in \mathbb{Z}_{\geq 2}$	Optimal objective function value for the BPPS
BPPS additional givens	**Description**
$\mathcal{S}^\star(I_c) \subseteq 2^{\mathcal{I}_c}$	Optimal solution for the BPP with items of class $c \in \mathcal{C}$ and bin capacity $d - s_c$
β_c	Optimal objective function 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 in $\mathcal{I}_c$ and bin capacity $d - s_c$
$\gamma_c = \lceil \sum_{i \in \mathcal{I}_c} w_i / (d - s_c) \rceil$	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

$\hat{k} = \sum_{c \in \mathcal{C}} \overline{\beta}_c$	Upper bound on the number of bins used in any optimal BPPS solution
$\check{k} = \lceil (\sum_{i \in \mathcal{I}} w_i \ + \ \sum_{c \in \mathcal{C}} \gamma_c \, s_c)/d \rceil$	Lower bound on the number of bins used in any optimal BPPS solution

Natural formulation for the BPPS	**Description**
ILP_{N}	Natural formulation for the BPPS
$\text{ILP}_{\text{N}}^{\dagger}$	ILP_{N} with MCIs
$\text{ILP}_{\text{N}}^{\ddagger}$	ILP_{N} with MCIs and MBI
$\text{ILP}_{\text{N}}^{\star}$	ILP_{N} with MCIs and MBI, and UB to the number of bins in any optimal solution

Arc-flow DAG notation	**Description**
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	Arc-flow DAG, with vertex set $\mathcal{V}$ and arc set $\mathcal{A}$
σ, τ	Source and sink vertices of $\mathcal{G}$
(ℓ, t, e)	Vertex of $\mathcal{G}$, identified by load $\ell \in \mathbb{Z}_{\geq 0}$, stage $t \in \mathbb{Z}_{\geq 0}$, and active flag $e \in \{-1, 0, 1\}$
$\omega_a \in \mathbb{Z}_{\geq 0}$	Weight of arc $a \in \mathcal{A}$
$\lambda_a \in \mathbb{Z}_{\geq 0}$	Cost of arc $a \in \mathcal{A}$
$\mathcal{A}_{\mathcal{I}}(i) \subseteq \mathcal{A}$	Set of item arcs associated with item $i \in \mathcal{I}$
$\mathcal{A}_{\mathcal{C}}(c) \subseteq \mathcal{A}$	Set of setup arcs associated with class $c \in \mathcal{C}$
$\phi(u, \tau)$	Sink label of $u \in \mathcal{V}$: weight of the maximum weight path from u to τ in $\mathcal{G}$
$\phi(\sigma, u)$	Source label of $u \in \mathcal{V}$: weight of the maximum weight path from σ to u in $\mathcal{G}$

Arc-flow formulation for the BPPS	**Description**
ILP_{AF}	Arc-flow formulation for the BPPS
$\text{ILP}_{\text{AF}}^{\dagger}$	ILP_{AF} with MCIs
$\text{ILP}_{\text{AF}}^{\star}$	ILP_{AF} with MCIs and MBI

Lower bounds on OPT	**Description**
$\zeta(\text{LP}_{\text{N}})$, $\zeta(\text{LP}_{\text{N}}^{\dagger})$, $\zeta(\text{LP}_{\text{N}}^{\ddagger})$, $\zeta(\text{LP}_{\text{N}}^{\star})$	Optimal objective function value of the LP relaxation of ILP_{N}, $\text{ILP}_{\text{N}}^{\dagger}$, $\text{ILP}_{\text{N}}^{\ddagger}$ and $\text{ILP}_{\text{N}}^{\star}$, respectively
$\zeta(\text{LP}_{\text{AF}})$, $\zeta(\text{LP}_{\text{AF}}^{\dagger})$, $\zeta(\text{LP}_{\text{AF}}^{\star})$	Optimal objective function value of the LP relaxation of ILP_{AF}, $\text{ILP}_{\text{AF}}^{\dagger}$ and $\text{ILP}_{\text{AF}}^{\star}$, respectively
$\varrho(LB) \in [0, 1]$	Worst-case performance ratio of a lower bound LB for the BPPS

E.2. Deferred proofs

E.2.1. A structural property of the BPP

An instance of the BPP is defined by a bin capacity d and a set of items $\mathcal{I} = \{1, 2, \ldots, n\}$, 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. In the literature (see, e.g., [Martello and Toth, 1990, Section 8.3.2]), the following inequality is known to hold for every BPP instance:

$$\left\lceil \frac{\sum_{i \in \mathcal{I}} w_i}{d} \right\rceil \ \geq \ \frac{\beta}{2}, \tag{E.1}$$

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*

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

Proof. Consider any optimal solution to the BPP. Let $\mathcal{S}^\star$ be the set of the β bins used in this optimal solution. Notice that, in an optimal BPP solution, there cannot exist two bins with total weight lower than or equal to $\frac{d}{2}$ 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

$$\sum_{i \in \mathcal{I}} w_i \; = \; \sum_{S \in \mathcal{S}^\star} \sum_{i \in S} w_i \; > \; \frac{\beta\, d}{2}.$$

Otherwise, there is at most one bin $\check{S} \in \mathcal{S}^\star$ with total load $\sum_{i \in \check{S}} w_i \leq \frac{d}{2}$. Since $\beta > 1$, there exists another bin $\hat{S} \in \mathcal{S}^\star$; by optimality, we must have

$$\sum_{i \in \check{S}} w_i + \sum_{i \in \hat{S}} w_i \; > \; d,$$

otherwise the two bins $\check{S}$ and $\hat{S}$ could be merged.

Hence,

$$\sum_{i \in \mathcal{I}} w_i = \sum_{\substack{i \in \mathcal{I}: \\ i \notin \check{S} \cup \hat{S}}} w_i + \sum_{i \in \check{S}} w_i + \sum_{i \in \hat{S}} w_i \; > \; \frac{d}{2}(\beta - 2) + d = \frac{\beta\, d}{2}.$$

□

E.2.2. Proof of Proposition 1

Proof. Multiplying the capacity constraints (2c) by the positive bin cost r and summing over all $b \in \mathcal{B}$ gives

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

By the assignment constraints (2b), $\sum_{b\in\mathcal{B}} x_{ib} = 1$ 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 (2d) imply

$$\sum_{b\in\mathcal{B}} y_{cb} \;\geq\; \sum_{b\in\mathcal{B}} x_{ib} \;=\; 1. \tag{E.3}$$

Therefore

$$r\sum_{i\in\mathcal{I}} w_i + r\sum_{c\in\mathcal{C}} s_c \;\leq\; d\sum_{b\in\mathcal{B}} r\, z_b, \quad\Longrightarrow\quad \sum_{b\in\mathcal{B}} r\, z_b \;\geq\; \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right). \tag{E.4}$$

Moreover,

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

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

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) \;\geq\; \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c \tag{E.5}$$

and, accordingly, the right-hand side of (E.5) is a lower bound on $\zeta(\mathrm{LP_N})$. Now consider the solution defined in (4). It satisfies constraints (3) by construction. Indeed, for every $i \in \mathcal{I}$,

$$\sum_{b\in\mathcal{B}} x_{ib} = \underbrace{k\,\frac{1}{k}}_{\text{by (4)}} = 1,$$

so the assignment constraints (2b) hold. Moreover, for each $b \in \mathcal{B}$,

$$\sum_{i\in\mathcal{I}} w_i\, x_{ib} + \sum_{c\in\mathcal{C}} s_c\, y_{cb} = \underbrace{\sum_{i\in\mathcal{I}} w_i\,\frac{1}{k}}_{\text{by (4)}} + \underbrace{\sum_{c\in\mathcal{C}} s_c\,\frac{1}{k}}_{\text{by (4)}} = \frac{1}{k}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) = \underbrace{d\, z_b}_{\text{by (4)}},$$

so the capacity constraints (2c) also hold at equality. Finally, the linking constraints (2d) are satisfied since by (4) the x-variables and y-variables take the same value $1/k$.

Hence, the solution defined in (4) is a feasible solution for $\mathrm{LP_N}$. Its objective value is

$$\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) = \underbrace{r\sum_{b\in\mathcal{B}} \frac{\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c}{k\, d}}_{\text{by (4)}} + \underbrace{\sum_{c\in\mathcal{C}} f_c \sum_{b\in\mathcal{B}} \frac{1}{k}}_{\text{by (4)}} = \frac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\right) + \sum_{c\in\mathcal{C}} f_c.$$

Thus, the objective value of the solution in (4) coincides with the right-hand side of (E.5). Consequently, the solution attains the lower bound and is therefore optimal, with optimal objective function value as in (5). □

E.2.3. Proof of Proposition 3

Proof. Consider a feasible solution to $\mathrm{ILP_N}$. From the assignment constraints (2b) and the linking constraints (2d), 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 $b \in \mathcal{B}_c$, since $z_b \leq 1$, the capacity constraints (2c) 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 $|\mathcal{B}_c|(d - s_c) = \big(\sum_{b \in \mathcal{B}} y_{cb}\big)(d - s_c)$, since $y_{cb} = 1$ only if $b \in \mathcal{B}_c$. Therefore, for every $c \in \mathcal{C}$, we have

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

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. □

E.2.4. 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 (E.3), we now directly rely on the MCI (6). It follows that every feasible

solution of $\mathrm{LP}_{\mathrm{N}}^{\dagger}$ satisfies

$$\sum_{b\in\mathcal{B}}\Big(r\,z_b+\sum_{c\in\mathcal{C}}f_c\,y_{cb}\Big)\;\geq\;\frac{r}{d}\left(\sum_{i\in\mathcal{I}}w_i+\sum_{c\in\mathcal{C}}\gamma_c\,s_c\right)\;+\;\sum_{c\in\mathcal{C}}\gamma_c\,f_c. \tag{E.6}$$

and, accordingly, the right-hand side of (E.6) is a lower bound on $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$. Now consider the solution defined in (7). 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

$$\begin{aligned}\sum_{b\in\mathcal{B}}\Big(r\,z_b+\sum_{c\in\mathcal{C}}f_c\,y_{cb}\Big) &= \underbrace{r\sum_{b\in\mathcal{B}}\frac{\sum_{i\in\mathcal{I}}w_i+\sum_{c\in\mathcal{C}}\gamma_c\,s_c}{k\,d}}_{\text{by (7)}}\;+\;\underbrace{\sum_{c\in\mathcal{C}}f_c\sum_{b\in\mathcal{B}}\frac{\gamma_c}{k}}_{\text{by (7)}}\\ &= \frac{r}{d}\left(\sum_{i\in\mathcal{I}}w_i+\sum_{c\in\mathcal{C}}\gamma_c\,s_c\right)+\sum_{c\in\mathcal{C}}\gamma_c\,f_c.\end{aligned}$$

Thus, the objective value of this solution coincides with the right-hand side of (E.6). Consequently, the solution attains the lower bound and is therefore optimal, with objective function value (8). □

E.2.5. Proof of Proposition 5

Proof. The proof relies on constructing a feasible BPPS solution through a three-step procedure, which we refer to as the *Constructive Heuristic Algorithm* (CHA) for the BPPS. The CHA constructs the three feasible solutions $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ whenever the stopping criteria for the corresponding steps are met. The proof considers the solution associated with the first applicable termination case and proves that $UB(\mathcal{S}) < 2\,\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$ for that case. Depending on the case, it may be easier to establish the inequality using a worse (higher-cost) solution; this is why we consider different possible terminations of the CHA.

Having a feasible BPPS solution $\mathcal{S}$ with $UB(\mathcal{S}) < 2\,\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger})$ suffices to prove the proposition. Indeed, the thesis is equivalent to

$$\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger}) > \frac{1}{2}\,OPT, \tag{E.7}$$

and, since every feasible BPPS solution $\mathcal{S}$ satisfies $UB(\mathcal{S}) \geq OPT$, to prove (E.7) it suffices to show that $\zeta(\mathrm{LP}_{\mathrm{N}}^{\dagger}) > \frac{1}{2}\,UB(\mathcal{S})$ for the feasible solution $\mathcal{S}$ returned by the CHA.

These are the three steps of the CHA:

1. For each class $c \in \mathcal{C}$, let I_c denote the associated BPP instance with item set $\mathcal{I}_c$, weights

$\{w_i\}_{i\in\mathcal{I}_c}$, and bin capacity $d_c := d - s_c$. In the first step of the CHA, for each class $c \in \mathcal{C}$ we solve the associated BPP instance I_c to optimality. We denote by $\mathcal{S}^\star(I_c)$ an optimal solution to I_c, and by $\beta_c := |\mathcal{S}^\star(I_c)|$ the corresponding optimal number of bins, i.e., the minimum number of bins of capacity $d_c = d - s_c$ required to pack all the items of class c. Let $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ denote the subset of classes (if any) for which $\beta_c = 1$, i.e., whose items can be packed into a single bin; for every $c \in \tilde{\mathcal{C}}$, $\mathcal{S}^\star(I_c) = \{\mathcal{I}_c\}$ is the corresponding single packing pattern. Moreover, since $\beta_c = 1$, $\sum_{i\in\mathcal{I}_c} w_i \leq d - s_c$; this yields $\gamma_c = 1$ for all $c \in \tilde{\mathcal{C}}$.

The outcome of this step is a feasible BPPS solution $\mathcal{S}_1 = \bigcup_{c\in\mathcal{C}} \mathcal{S}^\star(I_c)$, which contains, for every class $c \in \mathcal{C}$, exactly β_c bins whose items all belong to class c. If $\tilde{\mathcal{C}} = \emptyset$, i.e., if every class requires at least two bins, the CHA terminates at this point; otherwise, it proceeds to Step 2. The cost of the feasible solution $\mathcal{S}_1$ is

$$UB(\mathcal{S}_1) = \sum_{c\in\mathcal{C}} \beta_c\, f_c + r \sum_{c\in\mathcal{C}} \beta_c. \tag{E.8}$$

2. In the second step of the CHA, we define an auxiliary BPP instance $I_{\tilde{\mathcal{C}}}$ in which each class $c \in \tilde{\mathcal{C}}$ is represented by a single *class-aggregated item* of weight $\sum_{i\in\mathcal{I}_c} w_i + s_c$, and the bin capacity is d. We assume $I_{\tilde{\mathcal{C}}}$ is solved to optimality, and denote by $\mathcal{S}^\star(I_{\tilde{\mathcal{C}}})$ an optimal solution, and by $\delta := |\mathcal{S}^\star(I_{\tilde{\mathcal{C}}})|$ the corresponding optimal number of bins. If $\delta \geq 2$, i.e., if the class-aggregated items cannot all be packed into a single bin of capacity d, the outcome of this step is 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.

   The heuristic solution $\mathcal{S}_2$ obtained at this step 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, its cost is

$$UB(\mathcal{S}_2) = \sum_{c\in\mathcal{C}} \beta_c\, f_c + r\Big(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c \;+\; \delta\Big). \tag{E.9}$$

3. In the third step of the CHA, we create a single meta item, of weight $\sum_{c\in\tilde{\mathcal{C}}} \Big(\sum_{i\in\mathcal{I}_c} w_i + s_c\Big)$, corresponding to the total weight of the unique packing pattern assigned to a bin in Step 2 (this step is only reached when $\delta = 1$). Next, we select any class $c^\dagger \in \mathcal{C} \setminus \tilde{\mathcal{C}}$; such a class exists, since if $\mathcal{C} \setminus \tilde{\mathcal{C}} = \emptyset$ after Step 1 and $\delta = 1$, then all items in $\mathcal{I}$ fit into a single bin, which is the trivial case excluded in the Introduction to this paper. We then try to place the newly created

meta item into the residual capacity of one of the packing patterns in $\mathcal{S}^\star(I_{c^\dagger})$, say, the one with the minimum load. At the end of this step, there are two possible outcomes, both of which terminate the CHA. If the meta item fits into a packing pattern $S \in \mathcal{S}^\star(I_{c^\dagger})$, we obtain a feasible BPPS solution $\mathcal{S}_3$ which uses $\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c$ bins, with every class c still active in exactly β_c bins. Hence, its cost is

$$UB(\mathcal{S}_3) = \sum_{c\in\mathcal{C}} \beta_c\, f_c + r \sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c. \tag{E.10}$$

On the other hand, if the meta item does not fit into any packing pattern of $\mathcal{S}^\star(I_{c^\dagger})$, then the feasible solution $\mathcal{S}_3$ coincides with that from Step 2, and its cost (since here $\delta = 1$) is

$$UB(\mathcal{S}_3) = \sum_{c\in\mathcal{C}} \beta_c\, f_c + r\Big(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c \,+\, 1\Big). \tag{E.11}$$

We now show that, at every possible termination of the CHA, $\zeta(\mathrm{LP}^\dagger_{\mathrm{N}})$ is more than half of (E.8), (E.9), (E.10), or (E.11), depending on the respective termination.

Let us consider $\zeta(\mathrm{LP}^\dagger_{\mathrm{N}})$, computed as in (8):

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

which has a component of the form $\sum_{c\in\mathcal{C}} \gamma_c f_c$. We observe that all of (E.8), (E.9), (E.10), (E.11) 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 (E.1) applied to the BPP instance I_c, for every $c \in \mathcal{C}$.

Hence, to prove that $\zeta(\mathrm{LP}^\dagger_{\mathrm{N}}) > \frac{1}{2}\, UB(\mathcal{S})$, where $\mathcal{S}$ is a feasible BPPS solution returned by the CHA, it remains to show, depending on the termination of the CHA, the following four inequalities:

- $\dfrac{r}{d}\left(\displaystyle\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \dfrac{1}{2}\, r \displaystyle\sum_{c\in\mathcal{C}} \beta_c$ 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 by 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

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

$$= d \, \frac{\sum_{i \in \mathcal{I}_c} w_i}{d - s_c} > \frac{\beta_c \, d}{2} \tag{E.12}$$

where the last inequality comes from Lemma 1, applied to the BPP instance I_c, 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 by 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 (E.12), and

$$\sum_{c \in \tilde{\mathcal{C}}} \left(\sum_{i \in \mathcal{I}_c} w_i + s_c \right) > \frac{\delta \, d}{2} \tag{E.13}$$

is obtained by applying Lemma 1 to the BPP instance $I_{\tilde{\mathcal{C}}}$ 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 meta item fits into one of the packing patterns in $\mathcal{S}^\star(I_{c^\dagger})$, for some class $c^\dagger \in \mathcal{C} \setminus \tilde{\mathcal{C}}$. This inequality can be shown by 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 (E.12).

- $\dfrac{r}{d}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c\, s_c\right) > \dfrac{1}{2}\, r \left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c \,+\, 1\right)$ if the CHA terminates at the end of Step 3 and the meta item does not fit into any of the packing patterns in $\mathcal{S}^\star(I_{c^\dagger})$. This inequality can be shown by observing that

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

$$> \frac{1}{2}\sum_{\substack{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}\\ c\neq c^\dagger}} \beta_c \,+\, \frac{1}{2}\,(\beta_{c^\dagger}+1) = \frac{1}{2}\left(\sum_{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}} \beta_c \,+\, 1\right). \tag{E.14}$$

where, for every $c \in \mathcal{C} \setminus \tilde{\mathcal{C}}$ different from $c^\dagger$, we used (E.12).

Since $\mathcal{S}^\star(I_{c^\dagger})$ is an optimal solution to the BPP instance $I_{c^\dagger}$, no two of its $\beta_{c^\dagger}$ bins can be merged without exceeding the capacity $d_{c^\dagger}$, as otherwise $\mathcal{S}^\star(I_{c^\dagger})$ would not be optimal. Moreover, by assumption, the meta item – of weight $\sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right)$ – does not fit into the residual capacity of any bin in $\mathcal{S}^\star(I_{c^\dagger})$ either. Consider the auxiliary BPP instance with capacity $d - s_{c^\dagger}$, whose items are the $\beta_{c^\dagger}$ meta items obtained by aggregating the contents of each packing pattern in $\mathcal{S}^\star(I_{c^\dagger})$, plus one further item, namely the meta item of weight $\sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right)$. This auxiliary instance is either infeasible, if the additional meta item alone exceeds $d - s_{c^\dagger}$, or its optimal solution uses exactly $\beta_{c^\dagger}+1$ bins. Applying Lemma 1 — trivially, in the infeasible case — then yields

$$\sum_{i\in\mathcal{I}_{c^\dagger}} w_i + \sum_{c\in\tilde{\mathcal{C}}}\left(\sum_{i\in\mathcal{I}_c} w_i + s_c\right) > (d - s_{c^\dagger})\,\frac{\beta_{c^\dagger}+1}{2}.$$

Hence, to obtain (E.14) it suffices to show that

$$\frac{1}{d}\left(\sum_{\substack{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}\\ c\neq c^\dagger}} \left(\sum_{i\in\mathcal{I}_c} w_i + \gamma_c s_c\right) + \gamma_{c^\dagger} s_{c^\dagger} + (d - s_{c^\dagger})\,\frac{\beta_{c^\dagger}+1}{2}\right) \geq \frac{1}{2}\sum_{\substack{c\in\mathcal{C}\setminus\tilde{\mathcal{C}}\\ c\neq c^\dagger}} \beta_c \,+\, \frac{1}{2}\,(\beta_{c^\dagger}+1),$$

and therefore it is sufficient to show that

$$\gamma_{c^\dagger} \geq \frac{\beta_{c^\dagger} + 1}{2}.$$

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

□

E.2.6. Proof of Proposition 7

Proof. Consider any feasible solution $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ of $\mathrm{ILP_N}$. Summing the capacity constraints (2c) over all $b \in \mathcal{B}$ and using the assignment constraints gives

$$\sum_{i \in \mathcal{I}} w_i + \sum_{c \in \mathcal{C}} s_c \sum_{b \in \mathcal{B}} y_{cb} \leq d \sum_{b \in \mathcal{B}} z_b.$$

By Proposition 3, every integer-feasible solution of $\mathrm{ILP_N}$ satisfies the MCIs, and hence $\sum_{b \in \mathcal{B}} y_{cb} \geq \gamma_c$ for every $c \in \mathcal{C}$. Since $s_c \geq 0$, it follows that

$$\sum_{i \in \mathcal{I}} w_i + \sum_{c \in \mathcal{C}} \gamma_c s_c \leq d \sum_{b \in \mathcal{B}} z_b.$$

Therefore,

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

Finally, since the left-hand side is a sum of binary variables, the right-hand side can be rounded up, yielding (9). □

E.2.7. 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 (9) together with the MCIs (6). Hence, every feasible solution of $\mathrm{LP_N^\ddagger}$ satisfies

$$\sum_{b \in \mathcal{B}} \Big(r\, z_b + \sum_{c \in \mathcal{C}} f_c\, y_{cb} \Big) \;\geq\; r \sum_{b \in \mathcal{B}} z_b \;+\; \sum_{c \in \mathcal{C}} f_c \sum_{b \in \mathcal{B}} y_{cb} \;\geq\; r\, \check{k} \;+\; \sum_{c \in \mathcal{C}} \gamma_c\, f_c, \tag{E.15}$$

Accordingly, the right-hand side of (E.15) is a lower bound on $\zeta(\mathrm{LP_N^\ddagger})$.

Now consider the point defined in (10). For each $b \in \mathcal{B}$,

$$\sum_{i\in\mathcal{I}} w_i\, x_{ib} + \sum_{c\in\mathcal{C}} s_c\, y_{cb} = \underbrace{\sum_{i\in\mathcal{I}} w_i\, \frac{1}{k}}_{\text{by (10)}} + \underbrace{\sum_{c\in\mathcal{C}} s_c\, \frac{\gamma_c}{k}}_{\text{by (10)}} = \frac{1}{k}\left(\sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} \gamma_c s_c\right) \;\le\; \underbrace{\frac{d\,\check{k}}{k}}_{\text{by (9)}} = \underbrace{d\, z_b}_{\text{by (10)}},$$

so the capacity constraints (2c) 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 $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$, and is therefore a feasible solution. Its objective value is

$$\begin{aligned}\sum_{b\in\mathcal{B}}\Big(r\, z_b + \sum_{c\in\mathcal{C}} f_c\, y_{cb}\Big) &= \underbrace{r \sum_{b\in\mathcal{B}} \frac{\check{k}}{k}}_{\text{by (10)}} + \underbrace{\sum_{c\in\mathcal{C}} f_c \sum_{b\in\mathcal{B}} \frac{\gamma_c}{k}}_{\text{by (10)}} \\ &= r\,\check{k} + \sum_{c\in\mathcal{C}} \gamma_c\, f_c.\end{aligned}$$

Thus, the objective value of this solution coincides with the right-hand side of (E.15). Consequently, the solution attains the lower bound and is therefore optimal, with objective function value (11). □

E.2.8. Proof of Proposition 9

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

$$OPT = n\, r \quad \text{and} \quad \gamma_1 \;=\; \left\lceil \frac{n\,\vartheta}{2\,\vartheta - 1} \right\rceil.$$

Let $k = n$. By Proposition 8, the optimal objective function value of $\mathrm{LP}_{\mathrm{N}}^{\ddagger}$ is

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

Consider an even number of items n. Taking the limit of $\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger})$ as $\vartheta \to \infty$ gives

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

since, for all sufficiently large values of ϑ,

$$\left\lceil \frac{n}{2} + \frac{\left\lceil \frac{n\vartheta}{2\vartheta-1} \right\rceil}{2\vartheta} \right\rceil = \frac{n}{2} + 1.$$

Thus, we have that

$$\lim_{\vartheta \to \infty} \frac{\zeta(\mathrm{LP}_{\mathrm{N}}^{\ddagger})}{OPT} = \frac{r\left(\frac{n}{2}+1\right)}{rn} = \frac{1}{2} + \frac{1}{n},$$

which tends to $\frac{1}{2}$ for any sequence of instances with an even number of items tending to infinity. The result follows by combining this limit with Proposition 5.

□

E.2.9. Proof of Proposition 10

Proof. In any optimal BPPS solution $\mathcal{S}^{\star} \subseteq 2^{\mathcal{I}}$, we have

$$\underbrace{r\,|\mathcal{S}^{\star}| + \sum_{S \in \mathcal{S}^{\star}} \sum_{c \in \mathcal{C}(S)} f_c}_{=OPT} \;\leq\; r \sum_{c \in \mathcal{C}} \beta_c + \sum_{c \in \mathcal{C}} f_c\, \beta_c, \tag{E.16}$$

since the optimal objective function value of the BPPS, OPT, 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, since the items of any class $c \in \mathcal{C}$ cannot be packed into fewer than β_c bins, we have

$$\sum_{S \in \mathcal{S}^{\star}} \sum_{c \in \mathcal{C}(S)} f_c \;\geq\; \sum_{c \in \mathcal{C}} f_c\, \beta_c\,. \tag{E.17}$$

By chaining inequalities (E.16) and (E.17), we obtain

$$r\,|\mathcal{S}^{\star}| + \sum_{S \in \mathcal{S}^{\star}} \sum_{c \in \mathcal{C}(S)} f_c \leq r \sum_{c \in \mathcal{C}} \beta_c + \sum_{S \in \mathcal{S}^{\star}} \sum_{c \in \mathcal{C}(S)} f_c\,,$$

which implies

$$|\mathcal{S}^{\star}| \;\leq\; \sum_{c \in \mathcal{C}} \beta_c\,.$$

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 upper bound $\overline{\beta}_c \geq \beta_c$ still yields a valid (though possibly weaker) overall bound:

$$|\mathcal{S}^\star| \leq \sum_{c \in \mathcal{C}} \overline{\beta}_c.$$

□

E.3. Additional computational results

This section presents supplementary computational results that complement the analysis of Section 4.2. Specifically, Section E.3.1 reports the distribution of computing times and optimality gaps across the variants of ILP_N and ILP_AF via box plots. Section E.3.2 examines instead the quality of the LP relaxation bounds and the structural features of the optimal solutions across several instance classes.

E.3.1. Distribution of computing times and optimality gaps

This section reports the distributions of computation times and optimality gaps for the variants of ILP_N and ILP_AF across all tested instance sizes (in terms of the number of items n). Figures E.1 and E.2 show the distribution of computing times (in seconds, on a logarithmic scale) for the variants of ILP_N and ILP_AF, respectively. Each box plot reports the median (solid red line), the interquartile range, and the extreme values, together with the mean (dashed blue line) for the subset of instances sharing the same value of n. The box-plot data are restricted to the subset of instances solved to proven optimality by the strongest variant of each formulation, namely $\text{ILP}^\star_\text{N}$ and $\text{ILP}^\star_\text{AF}$, respectively.

For instances with $n = 25$, all variants in both families exhibit comparable median times, below 0.4 seconds, with $\text{ILP}^\star_\text{N}$ and $\text{ILP}^\star_\text{AF}$ already showing a lower mean due to fewer extreme outliers. The gap widens as n grows. For $n = 50$, the median time of $\text{ILP}^\star_\text{N}$ is approximately 0.14 seconds, roughly two orders of magnitude below that of ILP_N; within the arc-flow family, $\text{ILP}^\star_\text{AF}$ and $\text{ILP}^\dagger_\text{AF}$ achieve comparable medians below 6 seconds, while ILP_AF displays a substantially wider interquartile range and a higher mean. For $n = 100$, $\text{ILP}^\star_\text{N}$ reaches a median around 3 seconds, while $\text{ILP}^\star_\text{AF}$ achieves a median slightly above 50 seconds. At $n = 200$, ILP_N fails to solve the majority of instances within the time limit, while $\text{ILP}^\star_\text{N}$ maintains a median below 20 seconds. On the other hand, all the arc-flow

variants reach median times of hundreds of seconds for $n = 200$. Overall, the arc-flow variants tend to require more time than their natural-formulation counterparts across all values of n.

		ILP_N		$\text{ILP}_\text{N}^\dagger$		$\text{ILP}_\text{N}^\ddagger$		$\text{ILP}_\text{N}^\star$	
		time (s)		time (s)		time (s)		time (s)	
	inst	median	mean	median	mean	median	mean	median	mean
$n = 25$	137	0.4	41.7	0.3	62.0	0.1	25.3	0.0	20.8
$n = 50$	107	6.8	502.3	2.3	175.9	0.5	92.3	0.1	41.5
$n = 100$	79	607.2	920.5	28.2	439.3	9.6	169.5	3.3	103.7
$n = 200$	63	1,800.0	1,407.0	297.2	575.6	48.6	207.4	19.0	136.3

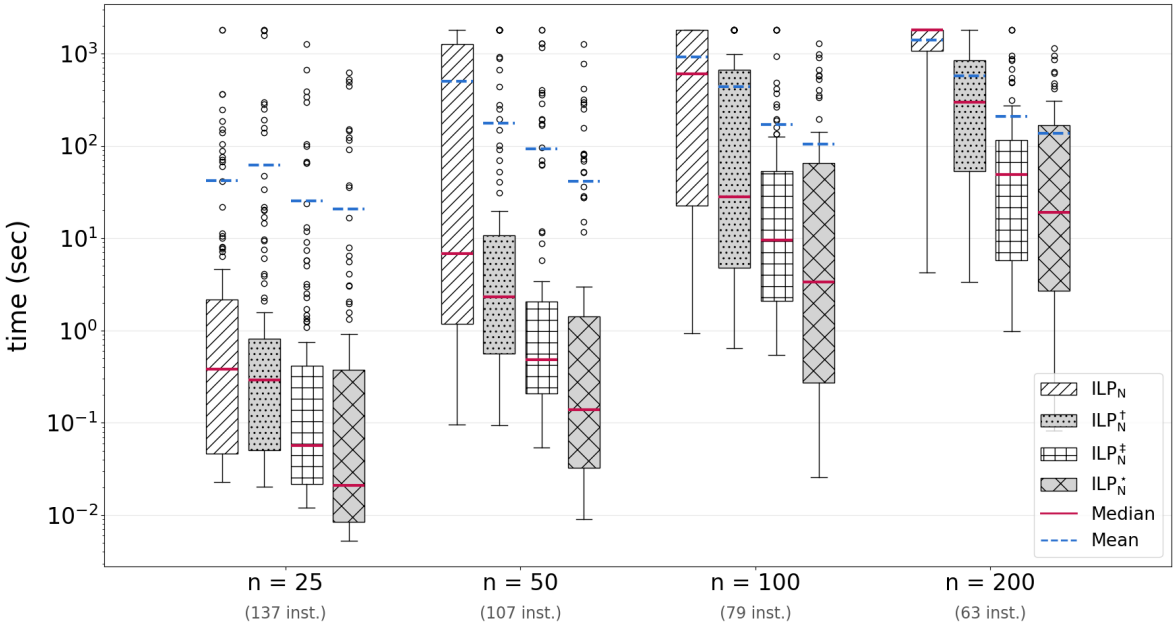


Figure E.1: Summary statistics (top) and box plots (bottom) of the solution times of the variants of ILP_N, restricted to instances solved to optimality by $\text{ILP}_\text{N}^\star$. Vertical axis on a logarithmic scale. Time limit: 1800 seconds.

Figures E.3 and E.4 show the distribution of the optimality gap (in percentage) for the variants of ILP_N and ILP_AF, respectively, restricted to the instances not solved to proven optimality by $\text{ILP}_\text{N}^\star$ and $\text{ILP}_\text{AF}^\star$ within the time limit. For the arc-flow family, only instances for which a valid lower bound was obtained for all variants within the time limit are included.

For the natural formulation, the results at $n = 25$ involve only seven instances and show little differentiation across variants, with median gaps all close to 11%. The picture changes substantially at larger scales. For $n = 50$, the median gap of ILP_N is 8.3%, while $\text{ILP}_\text{N}^\star$ reduces it to 5.9%; the gap between ILP_N and the variants that include the additional inequalities widens further as n grows. For $n = 100$, ILP_N achieves a median gap of 12.1%, whereas $\text{ILP}_\text{N}^\dagger$, $\text{ILP}_\text{N}^\ddagger$, and $\text{ILP}_\text{N}^\star$ all

		ILP_{AF}		$\text{ILP}^{\dagger}_{\text{AF}}$		$\text{ILP}^{\star}_{\text{AF}}$	
		time (s)		time (s)		time (s)	
	inst	median	mean	median	mean	median	mean
$n = 25$	144	0.2	14.5	0.2	22.3	0.2	11.0
$n = 50$	128	6.3	399.8	6.0	242.7	5.4	58.7
$n = 100$	69	157.5	754.9	94.7	649.7	53.8	242.6
$n = 200$	38	839.5	915.5	624.9	859.9	197.6	406.8

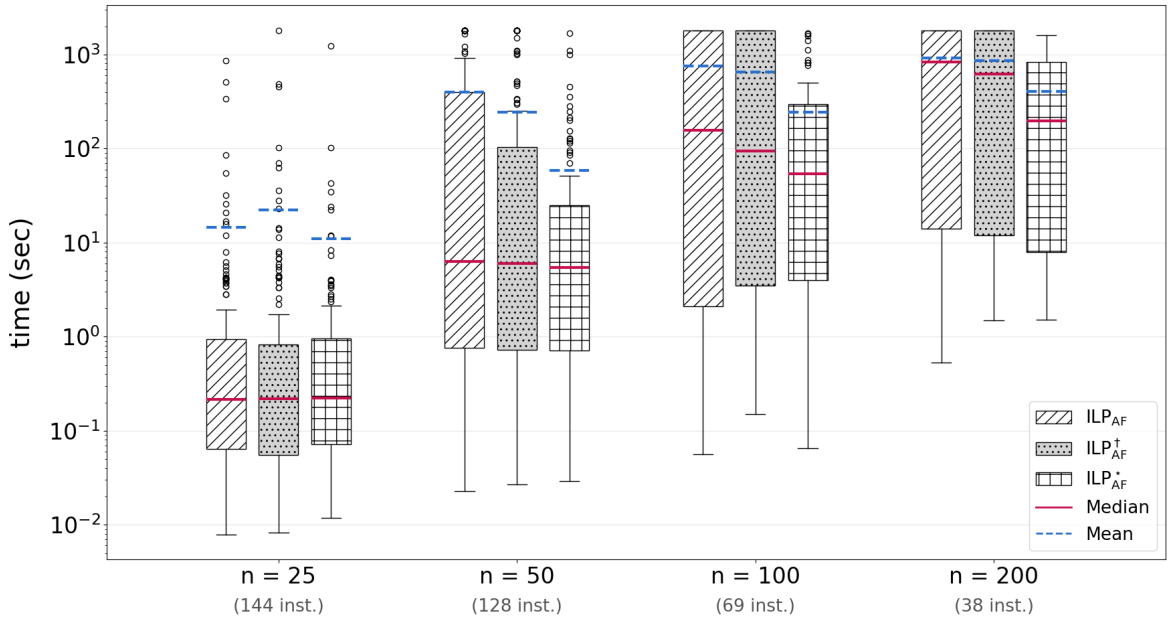


Figure E.2: Summary statistics (top) and box plots (bottom) of the solution times of the variants of ILP_{AF}, restricted to instances solved to optimality by $\text{ILP}^{\star}_{\text{AF}}$. Vertical axis on a logarithmic scale. Time limit: 1800 seconds.

attain a median near 5.3%, a reduction of more than half. The most pronounced differences appear at $n = 200$: ILP_{N} has a median gap of 12.7% and a mean of 16.4%, while $\text{ILP}^{\dagger}_{\text{N}}$, $\text{ILP}^{\ddagger}_{\text{N}}$, and $\text{ILP}^{\star}_{\text{N}}$ all maintain a median below 3.9% and a mean below 4.5%. These results indicate that the MCIs have a strong effect on bound quality, particularly on the hardest instances, and that the additional enhancements introduced in $\text{ILP}^{\ddagger}_{\text{N}}$ and $\text{ILP}^{\star}_{\text{N}}$ beyond those in $\text{ILP}^{\dagger}_{\text{N}}$ provide only marginal further improvement in the optimality gap.

For the arc-flow family, the number of unsolved instances available for this analysis is more limited, particularly at $n = 50$ (seven instances), so the corresponding statistics should be interpreted with caution. At $n = 100$, ILP_{AF} achieves a median gap of 5.6%, while $\text{ILP}^{\dagger}_{\text{AF}}$ and $\text{ILP}^{\star}_{\text{AF}}$ reduce this to approximately 4%. At $n = 200$, the three variants become nearly indistinguishable in terms of median gap, with values of 3.0%, 3.3%, and 2.7% for ILP_{AF}, $\text{ILP}^{\dagger}_{\text{AF}}$, and $\text{ILP}^{\star}_{\text{AF}}$, respectively,

		ILP_N		$\text{ILP}_\text{N}^\dagger$		$\text{ILP}_\text{N}^\ddagger$		$\text{ILP}_\text{N}^\star$	
		gap (%)		gap (%)		gap (%)		gap (%)	
	inst	median	mean	median	mean	median	mean	median	mean
$n = 25$	7	11.1	8.9	11.1	9.8	11.1	11.1	11.1	11.1
$n = 50$	37	8.3	8.8	6.3	7.3	6.2	6.6	5.9	6.7
$n = 100$	65	12.1	12.5	5.3	5.0	5.3	4.7	5.3	5.0
$n = 200$	81	12.7	16.4	3.8	4.4	3.6	4.3	3.8	4.5

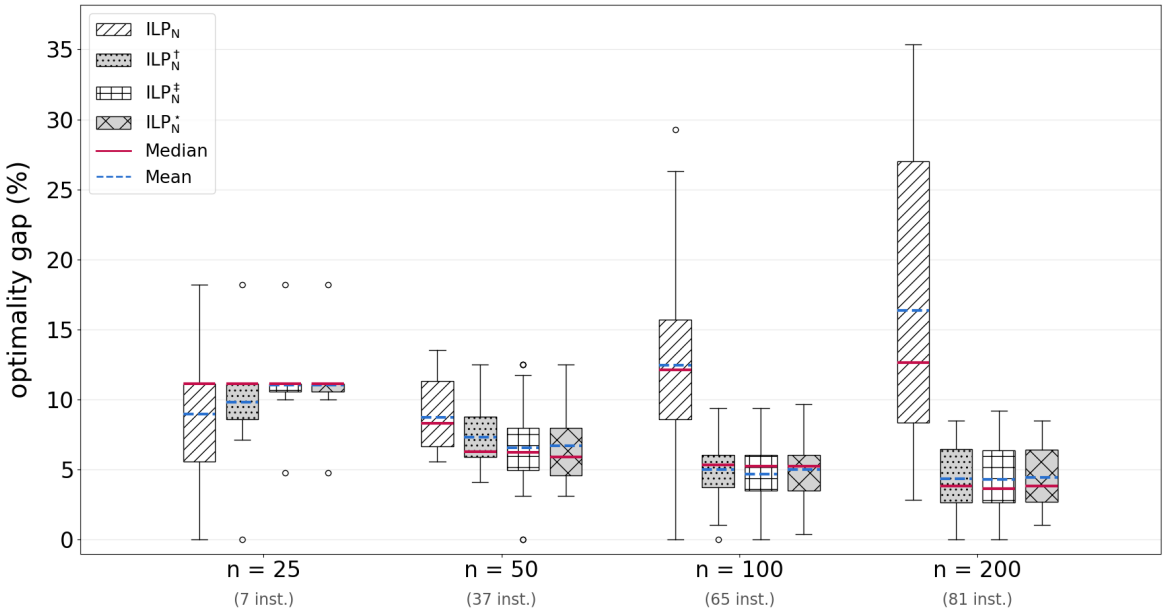


Figure E.3: Summary statistics (top) and box plots (bottom) of the optimality gaps of the variants of ILP_N, restricted to instances not solved to optimality by $\text{ILP}_\text{N}^\star$. Time limit: 1800 seconds.

suggesting that, on instances of this size, the valid inequalities provide limited additional bound improvement.

E.3.2. LP relaxation strength and optimal solution features: detailed results

In this section, we compare the LP relaxation lower bounds obtained with the variants of the ILP formulations we proposed in Sections 2 and 3, and analyze the main features of the optimal solutions obtained on our test set. The results are reported in Table E.2. The table is organized into two main parts. It reports aggregate statistics for the subset of instances with known optimum for which the LP relaxations of all tested formulations were solved to optimality within the time limit of 1800 seconds. This subset is chosen because it provides a reliable and consistent basis for comparing bound quality across formulations. The analysis covers a total of 427 random instances and 36 real-world instances, reported separately in the two panels of the table. Random instances are

		ILP_{AF}		$\text{ILP}^{\dagger}_{\text{AF}}$		$\text{ILP}^{\star}_{\text{AF}}$	
		gap (%)		gap (%)		gap (%)	
	inst	median	mean	median	mean	median	mean
$n = 50$	7	7.0	7.6	3.9	4.6	5.6	7.1
$n = 100$	39	5.6	5.0	3.9	3.7	4.0	3.7
$n = 200$	39	3.0	5.1	3.3	5.3	2.7	4.8

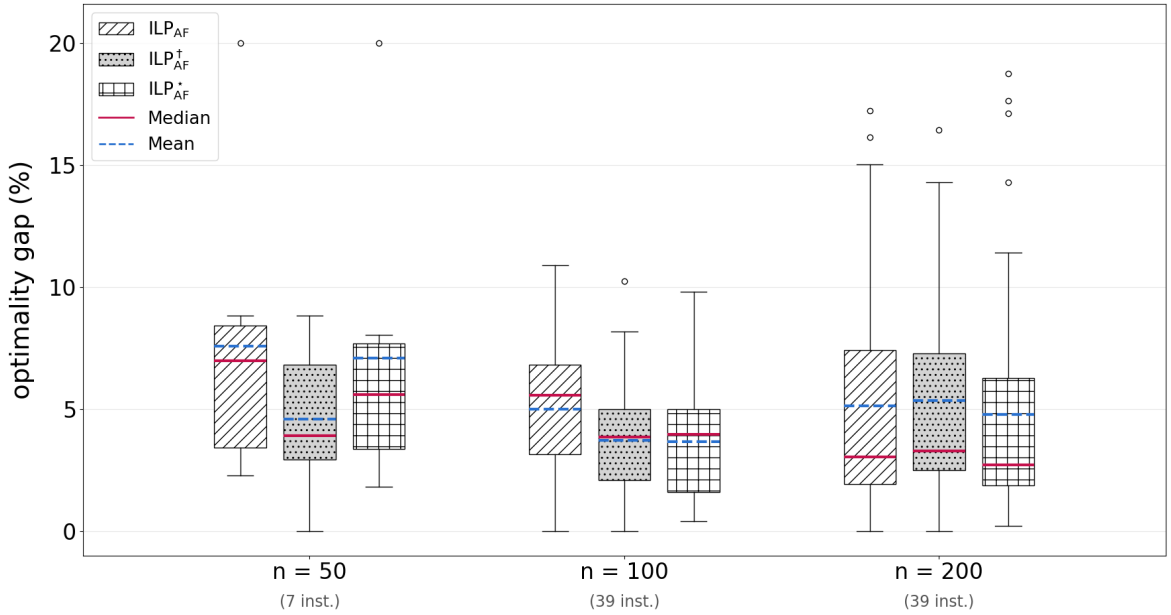


Figure E.4: Summary statistics (top) and box plots (bottom) of the optimality gaps of the variants of ILP_{AF}, restricted to instances not solved to optimality by $\text{ILP}^{\star}_{\text{AF}}$ and for which a valid dual bound was computed within the time limit for all the variants. Time limit: 1800 seconds.

grouped by different instance-generator parameter values, while real-world instances are grouped by the subset to which they belong.

The first part of Table E.2 (columns *Integrality gaps*) reports the integrality gap, defined as the percentage difference between the optimal objective function value of the BPPS and the optimal objective function value of the LP relaxation for a given formulation, taken with respect to the optimal objective function value of the BPPS. We consider the three natural formulations $\mathrm{ILP_N}$, $\mathrm{ILP_N^\dagger}$, and $\mathrm{ILP_N^\star}$, whose optimal LP relaxation objective values are $\zeta(\mathrm{LP_N})$, $\zeta(\mathrm{LP_N^\dagger})$, and $\zeta(\mathrm{LP_N^\star})$, respectively, as well as the three arc-flow formulations $\mathrm{ILP_{AF}}$, $\mathrm{ILP_{AF}^\dagger}$, and $\mathrm{ILP_{AF}^\star}$, whose optimal LP relaxation objective values are $\zeta(\mathrm{LP_{AF}})$, $\zeta(\mathrm{LP_{AF}^\dagger})$, and $\zeta(\mathrm{LP_{AF}^\star})$, respectively. As established in Equation (11), the upper bound $\hat{k}$ does not affect the optimal objective function value of the LP relaxation for $\mathrm{ILP_N^\star}$. Consequently, the optimal objective function 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 for the two proposed formulations and across different instance parameters. For the natural formulations, the results demonstrate that incorporating the MCIs significantly reduces the integrality gap across all problem settings compared to $\mathrm{LP_N}$. Formulation $\mathrm{LP_N^\star}$ achieves a further reduction when the MBI is added. On average over all random instances, the gap decreases from 17.8% for $\mathrm{LP_N}$ to 8.3% for $\mathrm{LP_N^\dagger}$ and 3.0% for $\mathrm{LP_N^\star}$. The arc-flow formulations achieve tighter bounds at each level of enhancement: the average gap is 6.6% for $\mathrm{LP_{AF}}$, 5.0% for $\mathrm{LP_{AF}^\dagger}$, and 1.6% for $\mathrm{LP_{AF}^\star}$. Furthermore, as n increases, the gap of the base natural formulation grows (15.6% for $\mathrm{LP_N}$ at $n = 25$ to 20.8% at $n = 200$), while those of the enhanced natural variants and of the arc-flow variants decrease substantially. For $n = 200$, $\mathrm{LP_{AF}^\star}$ achieves an average gap below 1%. Conversely, when the number of classes increases from $m = 5$ to $m = 10$, the gap of $\mathrm{LP_N}$ decreases while those of the enhanced natural variants do not change significantly; $\mathrm{LP_N^\star}$ and $\mathrm{LP_{AF}^\star}$ maintain gaps of 2.9% and 1.2%, respectively, at $m = 10$. Changes in bin capacity d have only a marginal effect on the integrality gaps across all formulations, suggesting that this parameter is less critical for bound quality. The presence of setup costs increases the gap for $\mathrm{LP_N}$ from 14.5% to 21.6%, whereas $\mathrm{LP_N^\star}$ stays around 3%. Among the arc-flow variants, the gap of $\mathrm{LP_{AF}}$ decreases slightly from 6.9% (no setup costs) to 6.3% (with setup costs), while $\mathrm{LP_{AF}^\star}$ achieves a gap of approximately 1.5% in both cases. This behavior for the natural formulations can be attributed to the fundamental difference in how the

Table E.2: Quality of the lower bound provided by the LP relaxation of the variants of the natural and arc-flow ILP formulations for the BPPS, and features of optimal BPPS solutions. The reported values refer to the subset of instances with a known optimum for which the LP relaxations of all tested formulations were solved to optimality within the 1800-second time limit.

			Integrality gap								Optimal BPPS solution features			
		#opt	LP_N	$LP_N^\dagger$	$LP_N^\star$	LP_{AF}	$LP_{AF}^\dagger$	$LP_{AF}^\star$	$\check{k}$	$\hat{k}$	#bins	#items	#classes	%fill
`item number`	$n = 25$	144	15.6	12.4	3.5	8.3	7.6	2.0	5.3	8.6	5.6	5.5	2.0	88.1
	$n = 50$	130	17.8	8.1	3.3	6.7	4.8	1.8	10.3	13.4	10.7	5.7	1.5	92.7
	$n = 100$	91	19.6	5.1	2.2	5.6	3.1	1.1	20.3	23.4	20.9	6.1	1.3	95.8
	$n = 200$	62	20.8	3.7	2.1	4.0	2.1	0.8	40.8	45.1	42.0	6.6	1.2	96.8
`class number`	$m = 5$	225	20.5	8.4	3.0	7.6	5.5	1.8	15.7	17.9	16.2	5.8	1.3	91.9
	$m = 10$	202	14.9	8.2	2.9	5.6	4.4	1.2	14.6	19.2	15.1	5.8	1.9	92.9
`bin capacity`	$d = 200$	174	17.9	7.6	2.2	6.7	4.9	1.1	16.0	19.3	16.4	6.7	1.5	93.1
	$d = 1{,}000$	141	18.1	8.9	3.8	6.7	5.4	2.1	14.5	17.9	15.2	5.6	1.6	91.7
	$d = 10{,}000$	112	17.5	8.5	3.2	6.3	4.8	1.7	14.7	18.2	15.3	4.8	1.7	92.1
`bin-setup costs`	no	224	14.5	9.4	2.9	6.9	6.3	1.6	15.9	19.3	16.4	5.8	1.7	93.6
	yes	203	21.6	7.1	3.1	6.3	3.5	1.5	14.4	17.6	14.9	5.9	1.5	91.1
`item weights`	small	90	15.3	11.9	1.8	11.1	9.4	1.4	5.3	9.2	5.4	11.6	2.3	88.9
	medium	107	16.4	6.5	1.3	6.6	4.5	0.9	13.0	15.8	13.1	5.2	1.4	93.8
	large	133	21.9	8.5	6.1	3.5	2.5	2.4	25.5	29.5	27.1	3.1	1.3	92.6
	mixed	97	16.3	6.6	1.6	6.7	4.8	1.3	12.4	15.2	12.5	4.9	1.5	93.8
`setup weights`	small	137	14.7	7.8	2.0	6.4	5.1	1.0	13.4	16.9	13.8	6.4	1.8	92.6
	large	151	20.4	8.8	4.0	6.7	4.9	2.1	16.6	19.8	17.3	5.4	1.4	91.9
	mixed	139	18.2	8.2	2.8	6.7	5.1	1.5	15.3	18.8	15.8	5.7	1.6	92.7
Total/Average		427	17.8	8.3	3.0	6.6	5.0	1.6	15.2	18.5	15.7	5.8	1.6	92.4
`real-world`	`rwA`	12	19.2	9.9	9.4	0.9	0.9	0.9	57.5	63.2	63.0	2.1	1.0	90.9
	`rwB`	12	22.7	14.7	14.1	0.4	0.4	0.4	59.8	70.7	69.8	1.9	1.0	85.2
	`rwC`	12	22.9	13.3	12.9	0.4	0.4	0.4	88.2	101.7	101.0	1.3	1.0	87.2
Total/Average		36	21.6	12.7	12.1	0.5	0.5	0.5	68.5	78.5	77.9	1.7	1.0	87.8

formulations handle class setup costs within their LP relaxation optimal objective function value. The optimal objective function value of LP_N accounts for each class exactly once (see Equation (5)), regardless of how many bins actually contain items of that class. In contrast, formulations $LP_N^\dagger$ and $LP_N^\star$, 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 Equation (8)). Consequently, when setup costs are present, the LP relaxation of ILP_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. Finally, item weights exhibit a notable interaction with the quality of the bound provided by the linear relaxations of both formulations. For instances with large item weights, the gap of $LP_N^\star$ rises to 6.1%, noticeably higher than for other weight categories. As far as setup weights are concerned,

small setup weights tend to reduce the gap for all the variants of $\mathrm{LP_N}$. The arc-flow formulation variants behave differently: setup weights do not seem to significantly influence the integrality gap, while all the arc-flow variants achieve small gaps on the instances with large-sized items, below 3.5% on average.

The second part of the table includes the average values of the lower bound $\check{k}$ and the upper bound $\hat{k}$, computed as detailed at the beginning of Section 4.2, as well as the structural characteristics of the optimal solutions found (*Optimal BPPS Solution Features*): these last columns report the average number of bins used in the optimal solution, the average number of items per bin, the average number of active classes per bin, and the average bin fill percentage. The effectiveness of the bounds can be assessed by comparing their average values to the actual average number of bins in optimal BPPS solutions. The lower bound $\check{k}$ demonstrates high accuracy across all parameter configurations, with average values differing from the actual bin count by at most 1.6 for instances with large-sized items, and approaching the optimal bin count in many subsets of instances. The upper bound $\hat{k}$, on the other hand, provides a looser estimate: for instances with $m = 10$, its average value is 19.2 while the actual average bin count is 15.1, a difference exceeding 25%. Despite this imprecision, $\hat{k}$ remains substantially smaller than the trivial bound n across all instance sizes, with the most significant reduction observed for larger instances (average $\hat{k} = 45.1$ vs. $n = 200$). This reduction enables a substantial decrease in the number of variables and constraints within formulation $\mathrm{ILP_N^\star}$, thereby contributing to its superior computational performance relative to the other $\mathrm{ILP_N}$ variants.

The solution statistics on random instances reveal that optimal solutions typically contain between 5 and 7 items per bin, involve a small number of active classes (around 1.6 on average), and achieve a high bin utilization rate (92.4% on average) across all parameter configurations. When the number of items n increases, the average number of items per bin grows while the number of active classes per bin decreases, reflecting a tendency towards more homogeneous bin composition in larger instances. Increasing the number of classes m results in slightly more diverse bins, whereas variations in bin capacity d have minimal impact on these statistics. The presence of setup costs is associated with marginally fewer active classes per bin, suggesting a packing strategy that limits costly setups. Instances with small item weights produce solutions with more items per bin (11.6 on average) and a higher class diversity (2.3 active classes per bin). In comparison, large item weights lead to fewer items per bin (3.1) and low class diversity (1.3).

For the real-world instances, the integrality gaps of the natural formulations remain large (21.6%

for $\mathrm{LP_N}$ and 12.1% for $\mathrm{LP_N^\star}$ on average), whereas the arc-flow formulations achieve gaps below 1% across all three instance groups. This contrast suggests that the arc-flow relaxation is particularly well-suited to the structure of real-world BPPS instances. The solution statistics reflect the different nature of these instances: optimal solutions use far more bins on average (up to 101 for family `rwC`), contain fewer items per bin (1.7 on average), and involve exactly one active class per bin in all groups of instances.