A Tight 2-Approximation Algorithm for the Bin Packing Problem with Setups

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

ARTICLE INFO

Keywords:
Bin packing with setups
Approximation algorithms
Worst-case analysis
Setup costs
Combinatorial optimization

ABSTRACT

We study approximation algorithms for the *Bin Packing Problem with Setups* (BPPS), a generalization of the classical *Bin Packing Problem* (BPP) in which items are partitioned into classes and activating a class in a bin consumes a setup weight and incurs a setup cost. We show that direct adaptations of Next Fit (NF), First Fit (FF), Best Fit (BF), and Worst Fit (WF), as well as their decreasing-order variants, have unbounded absolute worst-case performance ratios, even with unit-weight items and zero setup costs. We then introduce a two-phase algorithm, $\mathrm{TP}_{\mathcal{A}}$, that packs each class independently with a BPP algorithm $\mathcal{A}$ and subsequently merges compatible packing patterns. We prove that the solution returned by $\mathrm{TP}_{\mathcal{A}}$ has cost at most twice the optimum under the assumption that $\mathcal{A}$ produces pairwise merge-maximal solutions, i.e., such that no two packing patterns in the class-wise solution can be feasibly merged. If $\mathcal{A}$ also runs in polynomial time, this yields a 2-approximation algorithm for the BPPS. The factor is tight: the absolute worst-case performance ratio of $\mathrm{TP}_{\mathcal{A}}$ is exactly 2, even when $\mathcal{A}$ solves every class-wise BPP instance optimally. Since every Any Fit algorithm returns pairwise merge-maximal solutions, it follows that $\mathrm{TP}_{\mathrm{FF}}$, $\mathrm{TP}_{\mathrm{BF}}$, $\mathrm{TP}_{\mathrm{WF}}$, and their decreasing-order variants all have an absolute worst-case performance ratio exactly 2. If, in addition, $\mathcal{A}$ is an α-approximation algorithm with $\alpha \leq 2$, we obtain a finer, component-wise guarantee with factor 2 for the bin-opening cost and factor α for the setup-cost component.

1. Introduction

Consider an unlimited supply of identical *bins* of capacity $d \in \mathbb{Z}_{\geq 1}$, each incurring a fixed *bin cost* $r \in \mathbb{Z}_{\geq 1}$, and a set $\mathcal{I} = \{1, 2, \dots, n\}$ of n *items*, where each item $i \in \mathcal{I}$ has weight $w_i \in \mathbb{Z}_{\geq 1}$. The items are partitioned into m (nonempty) *classes*. For each class $c \in \mathcal{C} = \{1, 2, \dots, m\}$, let $\mathcal{I}_c \subseteq \mathcal{I}$ denote the set of items of class c. Each class $c \in \mathcal{C}$ is characterized by a *setup weight* $s_c \in \mathbb{Z}_{\geq 0}$ and a *setup cost* $f_c \in \mathbb{Z}_{\geq 0}$. If a bin contains at least one item of class c, then class c is said to be *active* in that bin, the available capacity of the bin is reduced by s_c, and an additional cost f_c is incurred. A bin is said to be *open* if it contains at least one item; each open bin incurs the fixed bin cost r.

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 the set of classes active in S, and let

$$\ell(S) := \sum_{i \in S} w_i + \sum_{c \in \mathcal{C}(S)} s_c$$

be its *load*. A *packing pattern* is a subset $S \subseteq \mathcal{I}$ such that $\ell(S) \leq d$. Thus, every packing pattern can be assigned to a single bin. A feasible solution to the *Bin Packing Problem with Setups* (BPPS) is a partition $\mathcal{S} \subseteq 2^{\mathcal{I}}$ of $\mathcal{I}$ into nonempty packing patterns. Its cost is

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

that is, the sum, over all packing patterns, of the cost of the bin to which the pattern is assigned and the setup costs of its active classes. The BPPS consists of finding a feasible solution of minimum cost. We denote by $OPT(I)$ the optimal

*Corresponding author
f.ciccarelli@uniroma1.it (F. Ciccarelli)
ORCID(s):

value of an instance I of the BPPS, i.e., the minimum total cost required to pack all items. Hence, for every feasible BPPS solution S to I, $UB(S) \geq OPT(I)$. 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., $w_i + s_c \leq d$. Integrality of the bin cost and of the setup costs is assumed only for ease of exposition: all the results of this paper hold for rational costs as well; only the item weights, the setup weights, and the bin capacity are required to be integer.

The BPPS is strongly NP-hard, as it admits the classical *Bin Packing Problem* (BPP) as a special case. The BPPS has applications in production planning and logistics, and was recently introduced in Baldacci, Ciccarelli, Coniglio, Dose and Furini (2025)—which, to the best of our knowledge, remains the only previous study devoted specifically to this problem. The authors of Baldacci et al. (2025) propose a natural integer linear programming formulation, as well as an arc-flow formulation, for the BPPS, analyze the linear programming relaxations of the two formulations, and derive strengthening valid inequalities. As far as exact algorithms for the classical BPP are concerned, the current state of the art is represented by branch-price-and-cut methods built on the set-partitioning (or set-covering) reformulation. Wei, Luo, Baldacci and Lim (2020) solve the pricing problem by an ad hoc label-setting algorithm and strengthen the formulation by means of subset-row inequalities; Baldacci, Coniglio, Cordeau and Furini (2024) combine column generation with a pattern-enumeration phase and numerically safe bounding techniques; and da Silva and Schouery (2025) propose an extended Ryan–Foster branching scheme together with a fast column generation procedure, improving on the previous best results for the closely related Cutting Stock Problem and for several of its packing relatives.

A problem closely related to the BPPS is the *Knapsack Problem with Setups* (KPS). In the KPS, items are partitioned into classes, and selecting any item of a class incurs both a setup cost and an additional capacity consumption. Thus, although the KPS asks for a maximum-profit subset of items that can be packed together rather than for packing all items into multiple bins, its class-activation structure mirrors that of the BPPS. We refer the reader to Michel, Perrot and Vanderbeck (2009), Furini, Monaci and Traversi (2018) and Pferschy and Scatamacchia (2018) for formulations, exact algorithms, and approximation results.

Several variants of the BPP are related to the BPPS. In the *Class-Constrained Bin Packing Problem* (CCBPP), items have colors and each bin may contain only a bounded number of colors. Shachnai and Tamir (2001) develop approximation schemes for class-constrained packing problems, whereas Shachnai and Tamir (2004) establish tight online bounds for the unit-size case. Xavier and Miyazawa (2008) study both offline and online algorithms for general item sizes, and Epstein, Imreh and Levin (2010) give an asymptotic fully-polynomial time approximation scheme for a constant number of classes, together with online results and inapproximability bounds. Exact algorithms for the CCBPP are instead developed by Borges, Miyazawa, Schouery and Xavier (2020). For the variable-sized version of the CCBPP, Dawande, Kalagnanam and Sethuraman (2001) proposed an asymptotic polynomial time approximation scheme and two 3-approximation algorithms; Xavier and Miyazawa (2008) provide a second asymptotic polynomial time approximation scheme when the number of classes is bounded by a constant.

Two further BPP variants are close in cost structure to the BPPS. Twigg and Xavier (2015) study a bicriteria problem that asks to minimize both the total number of bins and the number of bins spanned by the items of each color. In the *Bin Packing Problem with Minimum Color Fragmentation*, the number of available bins is instead fixed and the objective is to minimize the total number of times that colors appear in the bins (Bergman, Cardonha and Mehrani, 2019; Mehrani, Cardonha and Bergman, 2022; Barkel, Delorme, Malaguti and Monaci, 2025). With zero setup weights and unit setup costs, this objective coincides with the class-activation component of the BPPS objective; the BPPS additionally asks to choose how many bins to open and allows class-dependent setup weights and costs. Lastly, the price-of-clustering literature provides a further connection to the BPPS: it compares an optimal solution obtained by packing clusters of items independently with a globally-optimal packing (see, e.g., Epstein (2021, 2023)).

Despite the aforementioned streams of work, to the best of our knowledge, no approximation algorithm has been proposed specifically for the BPPS. Given the rich body of approximation results for the classical BPP—one of the most extensively studied problems in the approximation-algorithms literature, see Coffman, Csirik, Galambos, Martello and Vigo (2013); Galambos, Martello and Vigo (2025)—it is natural to ask whether the BPPS inherits some of its favorable approximability properties. In this paper, we take a first exploratory step in this direction and start investigating this question.

The remainder of the paper is organized as follows. Section 2 reviews the classical online and offline algorithms for the BPP, together with the performance measures and guarantees used throughout the article; it also reports the immediate consequence that, unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time algorithm for the BPPS can have an absolute worst-case performance ratio smaller than $3/2$. Section 3 studies the direct adaptations of these algorithms to the BPPS. We prove that classical BPP algorithms, as well as their decreasing-order variants, have unbounded worst-case

performance ratios if straightforwardly applied to the BPPS, even when all items have unit weight and all setup costs are zero. Section 4 introduces the two-phase algorithm $\mathrm{TP}_{\mathcal{A}}$, which first packs each class separately by means of a BPP algorithm $\mathcal{A}$ and then merges compatible packing patterns. We prove that the solution returned by $\mathrm{TP}_{\mathcal{A}}$ has cost at most twice the optimum whenever $\mathcal{A}$ produces pairwise merge-maximal class-wise packings, i.e., solutions in which no two packing patterns can be feasibly merged. We also show that this factor is tight, even if the class-wise subproblems are solved to optimality. It follows that, when $\mathcal{A}$ runs in polynomial time, $\mathrm{TP}_{\mathcal{A}}$ is a 2-approximation algorithm for the BPPS. Finally, we derive a refined guarantee that applies factor 2 to the bin-cost component of an optimal solution and the approximation factor of $\mathcal{A}$ to its setup-cost component. Section 5 concludes the paper and discusses directions for further research.

2. Classical BPP approximation algorithms

In this section, we review the classical online and offline approximation algorithms for the BPP and recall their performance guarantees. For a detailed review of them and their analyses, we refer the reader to Coffman et al. (2013); Galambos et al. (2025). These algorithms provide both the benchmark for the direct BPPS adaptations studied in Section 3 and for the algorithms used in Phase 1 of the two-phase algorithm introduced in Section 4. Before describing them, we introduce the notation needed to state their approximation properties. In particular, we recall the notions of an α-approximation algorithm and absolute worst-case performance ratio.

Let $\mathcal{A}$ be an algorithm for a minimization problem. For any instance I, denote by $\mathcal{A}(I)$ the value of the solution returned by $\mathcal{A}$ and by $OPT(I)$ the optimal value (assumed to be positive). We assume that $\mathcal{A}$ always returns a feasible solution; hence, for every instance I, it holds that $\mathcal{A}(I) \geq OPT(I)$ and $\mathcal{A}(I)/OPT(I) \geq 1$. If there exists a finite constant $\alpha \in [1, +\infty)$ such that

$$\frac{\mathcal{A}(I)}{OPT(I)} \leq \alpha \quad \text{for all instances } I,$$

and, in addition, $\mathcal{A}$ runs in polynomial time, then $\mathcal{A}$ is called an *α-approximation algorithm*.

Furthermore, the *absolute worst-case performance ratio* of $\mathcal{A}$ is

$$R_{\mathcal{A}} := \sup_{I} \left\{ \frac{\mathcal{A}(I)}{OPT(I)} \right\} \in [1, +\infty]. \tag{2}$$

Hence, a polynomial time algorithm $\mathcal{A}$ is an α-approximation algorithm if and only if $R_{\mathcal{A}} \leq \alpha$. Nonetheless, note that $R_{\mathcal{A}}$ is defined for every algorithm $\mathcal{A}$, including non-polynomial ones. A factor α such that $\mathcal{A}(I) \leq \alpha\, OPT(I)$ for all I is called *tight* if it coincides with the ratio in (2), i.e., if $R_{\mathcal{A}} = \alpha$, so that α cannot be replaced by any smaller value.

Before reviewing the classical algorithms, we comment on which performance measure is meaningful for the BPPS. In the BPP literature it is customary to complement the absolute worst-case performance ratio with its asymptotic counterpart, obtained by restricting the supremum in (2) to instances whose optimal value is large. For the BPPS this is vacuous, as the following remark shows.

Remark 2.1. *Call an algorithm for the BPPS* cost-oblivious *if the partition it returns depends on the instance only through $\mathcal{I}$, C, d, $\boldsymbol{w}$ and $\boldsymbol{s}$, and not through the bin cost r or the setup costs $\boldsymbol{f}$. Every algorithm considered in this paper is cost-oblivious. Given a BPPS instance $I = (\mathcal{I}, C, d, r, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{f})$ and $\lambda \in \mathbb{Z}_{\geq 1}$, let $I(\lambda) := (\mathcal{I}, C, d, \lambda r, \boldsymbol{w}, \boldsymbol{s}, \lambda \boldsymbol{f})$. The two instances have the same feasible solutions, and by (1) the cost of each of them is multiplied by λ in $I(\lambda)$. Hence $OPT(I(\lambda)) = \lambda\, OPT(I)$ and, for a cost-oblivious algorithm $\mathcal{A}$, $\mathcal{A}(I(\lambda)) = \lambda\, \mathcal{A}(I)$. Two consequences follow.*

- *(i) The ratio $\mathcal{A}(I)/OPT(I)$ is invariant under cost scaling, while $OPT(I(\lambda)) \to +\infty$ as $\lambda \to +\infty$. Restricting the supremum in (2) to instances with large optimal value would therefore return $R_{\mathcal{A}}$ itself. This is in contrast with the BPP, where the optimal value is the number of open bins and is thus a purely combinatorial quantity, so that letting it grow does constrain the structure of the instance;*
- *(ii) Additive refinements are impossible: if $\mathcal{A}(I) \leq \rho\, OPT(I) + \kappa$ for every instance I, for some constants $\rho \geq 1$ and $\kappa \geq 0$, then applying this inequality to $I(\lambda)$, dividing by λ and letting $\lambda \to +\infty$ yields $\mathcal{A}(I) \leq \rho\, OPT(I)$ for every I. In particular, no BPPS counterpart of bounds such as the one recalled below for FFD, in which an additive term buys a smaller multiplicative factor, can hold for a cost-oblivious algorithm.*

For these reasons, all the results of this paper are stated in terms of the absolute worst-case performance ratio. A non-degenerate asymptotic notion, obtained by letting the number of open bins grow, is discussed in Remark 3.4 below.

We now briefly review classical *online algorithms* for the BPP, i.e., algorithms that process the items in the given order of the instance.

- *Next Fit* (NF) maintains exactly one open bin and places the current item into it if it fits; otherwise, it closes the bin and opens a new one.
- *First Fit* (FF) places the current item into the lowest-indexed open bin into which it fits, opening a new bin if necessary.
- *Best Fit* (BF) places the current item into an open bin that leaves the minimum residual capacity (equivalently, it chooses a bin with enough residual capacity having maximum load), opening a new bin if necessary.
- *Worst Fit* (WF) places the current item into a feasible open bin that leaves the maximum residual capacity (equivalently, it chooses a bin with enough residual capacity having minimum load), opening a new bin if necessary.

FF, BF, and WF are *Any Fit* algorithms, i.e., algorithms that open a new bin only when the current item does not fit in any of the previously opened bins; see Johnson (1973). These online algorithms are also commonly used in the *offline* setting by first sorting the items in non-increasing order of weight and then applying the same packing rule. In this case, they are called *Next Fit Decreasing* (NFD), *First Fit Decreasing* (FFD), *Best Fit Decreasing* (BFD), and *Worst Fit Decreasing* (WFD). Throughout the paper, we assume that items of equal weight are sorted according to their relative order in the input instance. This tie-breaking convention is part of the definition of every decreasing-order algorithm considered here.

All the algorithms mentioned above run in polynomial time and are α-approximation algorithms for the BPP; the corresponding values of α are recalled below. We refer the reader to the surveys Coffman et al. (2013); Galambos et al. (2025) for further details and for an overview of other approximation algorithms for the BPP. In particular, every Any Fit algorithm has an absolute worst-case performance ratio at most 2, and both WF and NF have an absolute worst-case performance ratio exactly 2; see Boyar, Dósa and Epstein (2012) for the exact absolute results and Johnson, Demers, Ullman, Garey and Graham (1974) for the classical worst-case analysis. For FF and BF, the tight absolute bounds

$$\mathrm{FF}(I) \le \left\lfloor \frac{17\,OPT(I)}{10} \right\rfloor \qquad \text{and} \qquad \mathrm{BF}(I) \le \left\lfloor \frac{17\,OPT(I)}{10} \right\rfloor$$

were established by Dósa and Sgall (2013) and by Dósa and Sgall (2014), respectively; thus, $R_{\mathrm{FF}} = R_{\mathrm{BF}} = 17/10$. For their decreasing variants, we instead have

$$R_{\mathrm{FFD}} = R_{\mathrm{BFD}} = \frac{3}{2}\,.$$

These equalities follow from Simchi-Levi (1994). Moreover, no polynomial-time algorithm for the BPP can achieve an absolute worst-case performance ratio strictly smaller than $3/2$ unless $\mathsf{P} = \mathsf{NP}$; this is a classical consequence of the NP-completeness of PARTITION, see Garey and Johnson (1979). The tight additive refinement due to Dósa, Li, Han and Tuza (2013) is $\mathrm{FFD}(I) \le 11/9\,OPT(I) + 6/9$. Since no later result in this paper uses the finer classical bounds for NFD or WFD, we do not report them here.

Since the BPP is the special case of the BPPS in which a single class with zero setup weight and zero setup cost is present, the $3/2$ threshold recalled above transfers verbatim to the BPPS.

Proposition 2.2. *Unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time algorithm for the BPPS has an absolute worst-case performance ratio smaller than $3/2$.*

Proof. Consider any BPP instance with bin capacity d, item set $\mathcal{I}$ and item weights $\boldsymbol{w}$, and let I be the BPPS instance with the same items and the same bin capacity, a single class $C = \{1\}$ with $\mathcal{I}_1 = \mathcal{I}$, setup weight $s_1 = 0$, setup cost $f_1 = 0$, and bin cost $r = 1$. Since $s_1 = 0$, the standing assumption $w_i + s_1 \le d$ reduces to $w_i \le d$ for all $i \in \mathcal{I}$, which holds for every feasible BPP instance. The two instances have the same feasible solutions, and $UB(S) = |S|$ for every

such solution S by (1). Hence $OPT(I)$ equals the minimum number of bins required by the original BPP instance. The transformation is computable in linear time, so any polynomial-time algorithm for the BPPS with absolute worst-case performance ratio ρ yields, by composition, a polynomial-time algorithm for the BPP with the same ratio. The claim thus follows from the $3/2$ threshold recalled above. □

3. Poor worst-case performance of classical BPP algorithms on BPPS instances

In this section, we show that straightforward adaptations of the classical BPP algorithms NF, FF, BF, and WF have no worst-case guarantee on BPPS instances. The same holds for their decreasing-order variants NFD, FFD, BFD, and WFD. For the BPPS, these algorithms process items in the given order and iteratively apply their packing rule (next, first feasible, best feasible, or worst feasible, respectively), accounting for class setup weights in the residual-capacity calculation: when an item is packed into a bin whose class is not yet active, the corresponding setup cost is incurred, and the setup weight is charged once for that bin, reducing the remaining capacity by the setup weight in addition to the item weight.

To prove that these algorithms have no worst-case guarantee for the BPPS, we propose families of instances that isolate the effect of setup weights by assigning unit weight to every item and setting all setup costs to zero. An optimal solution to these instances consists of one packing pattern for the items of each class; the solutions returned by straightforward extensions of the BPP algorithms to the BPPS instead repeatedly activate the same classes, thereby consuming a noticeable amount of capacity due to the setup weights. Since the number of these class activations grows with the number of items while the optimal packing opens a constant number of bins, the resulting ratio is unbounded. Thanks to employing identical item weights, the same families of instances are directly applicable to the decreasing-order variants.

We start with NF and exhibit a family of BPPS instances on which NF generates one packing pattern per item, whereas an optimal BPPS solution consists of only two packing patterns.

Proposition 3.1. *For the* BPPS*, the absolute worst-case performance ratio of* NF *is unbounded.*

Proof. Fix an even $n \geq 4$ and consider the BPPS instance I_n with $m = 2$ classes, $|\mathcal{I}_1| = |\mathcal{I}_2| = n/2$, and

$$d = n - 1, \qquad w_i = 1, \;\; i \in \mathcal{I}, \qquad s_1 = s_2 = \frac{n}{2} - 1, \qquad f_1 = f_2 = 0.$$

For each $c \in \{1, 2\}$,

$$\sum_{i \in \mathcal{I}_c} w_i + s_c = \frac{n}{2} + \left(\frac{n}{2} - 1\right) = n - 1 = d.$$

Thus, the items of each class form a feasible packing pattern. Moreover, no packing pattern can contain items from both classes, since $s_1 + s_2 + 2 = n > d$. Hence $OPT(I_n) = 2\,r$.

Assume the items are ordered so that their class labels alternate as $1, 2, 1, 2, \ldots$. After packing the first item (class 1), the residual capacity is

$$d - (1 + s_1) = (n - 1) - \frac{n}{2} = \frac{n}{2} - 1.$$

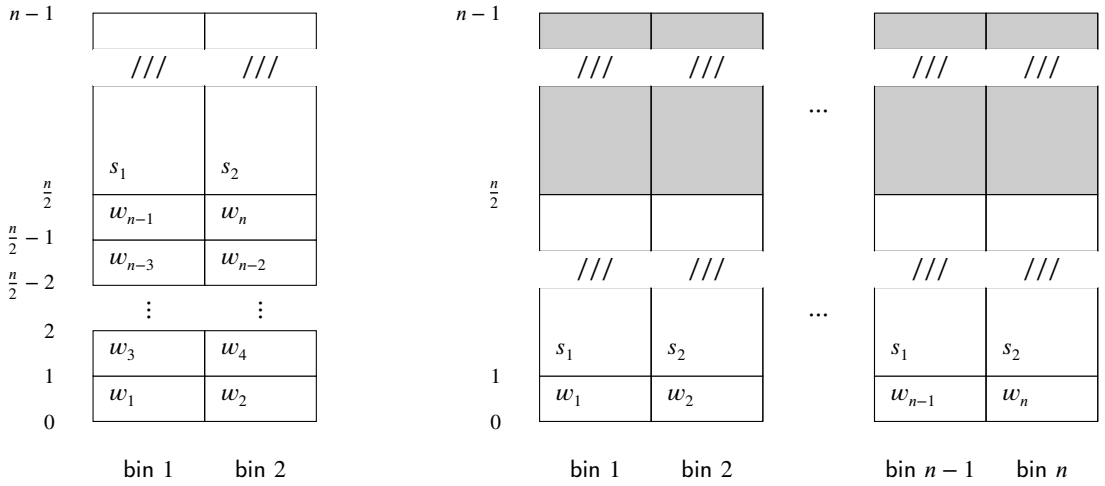
The next item (class 2) would require $1 + s_2 = n/2 > n/2 - 1$ units of capacity, so NF opens a new bin. The same argument repeats for every item, and thus $\mathrm{NF}(I_n) = n\,r$. Therefore,

$$\frac{\mathrm{NF}(I_n)}{OPT(I_n)} = \frac{n\,r}{2\,r} = \frac{n}{2} \xrightarrow[n \to \infty]{} +\infty.$$

□

An illustration of the difference between an optimal BPPS solution and the solution found by NF for the family of instances in the proof of Proposition 3.1 is given in Figure 1.

We continue with FF, BF, and WF, describing a family of BPPS instances on which these algorithms open a number of bins which grows linearly with the number of items, whereas an optimal BPPS solution consists of only three bins.



(a) An optimal BPPS solution

(b) The NF algorithm solution

Figure 1: Worst-case family of instances for NF used in the proof of Proposition 3.1. The items are ordered as done in the proof. Blocks represent item weights and the setup weight of their class; the shaded areas indicate unused capacity. Left: an optimal BPPS solution with two packing patterns, one per class, each assigned to a bin. Right: the NF solution with one packing pattern per item, each assigned to a bin filled up to $n/2$ by a single item plus its class setup weight.

Proposition 3.2. *For the* BPPS*, the absolute worst-case performance ratios of* FF*,* BF*, and* WF *are unbounded.*

Proof. Let $n \geq 12$ be divisible by 6 (the present proof only requires $n \geq 6$; the stronger assumption $n \geq 12$ is used in Remark 3.4) and consider the BPPS instance I_n with $m = 3$ classes, $|\mathcal{I}_1| = |\mathcal{I}_2| = |\mathcal{I}_3| = n/3$, and

$$d = \frac{2n}{3} - 1, \qquad w_i = 1, \;\; i \in \mathcal{I}, \qquad s_1 = s_2 = \frac{n}{3} - 1, \qquad s_3 = \frac{n}{3} - 2, \qquad f_c = 0, \;\; c \in C.$$

The items of each class form one packing pattern since

$$\sum_{i \in \mathcal{I}_1} w_i + s_1 = \sum_{i \in \mathcal{I}_2} w_i + s_2 = \frac{2n}{3} - 1 = d, \qquad \sum_{i \in \mathcal{I}_3} w_i + s_3 = \frac{2n}{3} - 2 < d.$$

Hence, $OPT(I_n) \leq 3\,r$. Moreover, no bin can contain items of both classes 1 and 2, because even packing only one item of each would require

$$s_1 + s_2 + 2 = \left(\frac{n}{3} - 1\right) + \left(\frac{n}{3} - 1\right) + 2 = \frac{2n}{3} > d.$$

Thus, items of classes 1 and 2 must belong to different bins; with only two open bins, they would each be full, leaving no capacity for any item of class 3. Hence $OPT(I_n) \geq 3\,r$, and therefore $OPT(I_n) = 3\,r$.

Assume the items are ordered so that their class labels follow the sequence $(1, 2, 3, 3)$, repeated exactly $n/6$ times, followed by the remaining $n/6$ items of class 1 and then the remaining $n/6$ items of class 2. When processing each repetition of the sequence, FF, BF and WF open a new bin for the class-1 item and a new bin for the class-2 item since, after packing any of these items, the residual capacity is

$$d - (1 + s_1) = d - (1 + s_2) = \left(\frac{2n}{3} - 1\right) - \frac{n}{3} = \frac{n}{3} - 1,$$

while an item of the other class would require a capacity of $1 + s_2 = 1 + s_1 = \frac{n}{3} > \frac{n}{3} - 1$. Each class-3 item requires $1 + s_3 = \frac{n}{3} - 1$, so the two class-3 items in the pattern completely fill up the residual space of the two bins just opened; the two bins are indistinguishable for FF, BF and WF, so the argument does not depend on how ties are broken. Hence, when processing each repetition of the sequence, the algorithm opens two new bins.

After $n/6$ repetitions of the sequence, $n/3$ bins have been opened and are full, and all class-3 items are packed. The remaining $n/6$ items of class 1 fit in one additional bin, and the same holds for those of class 2, since

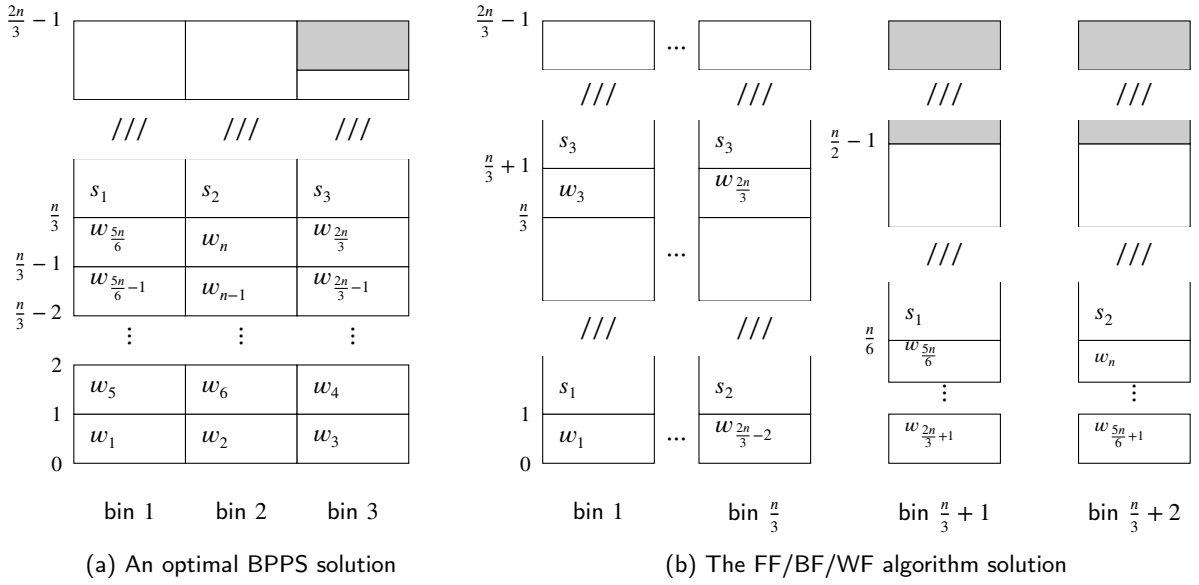
$$\frac{n}{6} + s_1 = \frac{n}{6} + s_2 = \frac{n}{6} + \left(\frac{n}{3} - 1\right) = \frac{n}{2} - 1 \leq d.$$

Therefore $\mathrm{FF}(I_n) = \mathrm{BF}(I_n) = \mathrm{WF}(I_n) = \left(\frac{n}{3} + 2\right) r$, and

$$\frac{\mathrm{FF}(I_n)}{OPT(I_n)} = \frac{\mathrm{BF}(I_n)}{OPT(I_n)} = \frac{\mathrm{WF}(I_n)}{OPT(I_n)} = \frac{\left(\frac{n}{3} + 2\right) r}{3r} = \frac{n}{9} + \frac{2}{3} \xrightarrow[n\to\infty]{} +\infty.$$

□

An illustration of the difference between an optimal BPPS solution and the solution found by FF, BF, and WF for the family of instances in the proof of Proposition 3.2 is given in Figure 2.



(a) An optimal BPPS solution

(b) The FF/BF/WF algorithm solution

Figure 2: Worst-case family of instances for FF, BF, and WF used in the proof of Proposition 3.2. The items are ordered as indicated in the proof. Left: an optimal BPPS solution has three packing patterns, one per class. Right: during each of the first $n/6$ repetitions of the $(1, 2, 3, 3)$ sequence, FF, BF, and WF generate one new packing pattern for the class-1 item and one for the class-2 item. The two class-3 items fill up the residual capacities of the associated bins. The remaining $n/6$ items of class 1 and the remaining $n/6$ items of class 2 are then packed into one additional bin per class.

Note that in the instance families considered in the proofs of Propositions 3.1 and 3.2 all items have unit weight. Therefore, sorting them by non-increasing weight does not induce any meaningful order. Under the tie-breaking convention introduced in Section 2, equal-weight items retain the input order used in the two constructions. Hence, the results shown for NF, FF, BF, and WF also hold for the decreasing-order variants NFD, FFD, BFD, and WFD. Thus:

Corollary 3.3. *For the* BPPS*, the absolute worst-case performance ratios of* NFD*,* FFD*,* BFD*, and* WFD *are unbounded.*

Remark 3.4. *The families used in Propositions 3.1 and 3.2 have a bounded optimal value, equal to* $2r$ *and* $3r$ *respectively, so the two statements concern the absolute worst-case performance ratio. By Remark 2.1, no stronger conclusion can be drawn from an asymptotic notion based on letting* $OPT(I)$ *grow. A non-degenerate asymptotic measure is instead obtained by letting the* number of open bins *grow, which is the exact counterpart of the classical BPP definition. Let*

$\nu(I)$ denote the minimum number of packing patterns over all optimal solutions to a BPPS instance I; the minimum is needed because, when setup costs are positive, distinct optimal solutions may use a different number of bins. Set

$$R_{\mathcal{A}}^{\infty} := \lim_{k\to+\infty} \sup_{I} \left\{ \frac{\mathcal{A}(I)}{OPT(I)} \ : \ \nu(I) \geq k \right\} \ \leq \ R_{\mathcal{A}}.$$

Propositions 3.1 and 3.2, being lower bounds on $R_{\mathcal{A}}$, do not transfer to $R_{\mathcal{A}}^{\infty}$ by themselves. They do, however, extend by replication. For $\vartheta \in \mathbb{Z}_{\geq 1}$, let I_n^{ϑ} be the instance obtained by juxtaposing ϑ copies of I_n, each copy having its own classes, and by listing the items copy after copy. When a copy starts, no bin opened while processing the previous ones has enough residual capacity to activate a class of the new copy: in the family of Proposition 3.1 the only open bin has residual capacity $\frac{n}{2} - 1 < \frac{n}{2} = 1 + s_c$, whereas in that of Proposition 3.2 every previously opened bin is either full or has residual capacity $\frac{n}{6} < \frac{n}{3} - 1 \leq 1 + s_c$, which is where $n \geq 12$ is used. Hence each algorithm processes every copy exactly as it processes I_n, giving $\mathcal{A}(I_n^{\vartheta}) = \vartheta\, \mathcal{A}(I_n)$, while $OPT(I_n^{\vartheta}) \leq \vartheta\, OPT(I_n)$ because the union of ϑ optimal solutions, one per copy, is feasible. Therefore

$$\frac{\mathcal{A}(I_n^{\vartheta})}{OPT(I_n^{\vartheta})} \ \geq \ \frac{\mathcal{A}(I_n)}{OPT(I_n)},$$

Moreover, I_n^{ϑ} contains ϑn unit-weight items and has the same bin capacity d as I_n, so that every feasible solution to I_n^{ϑ} uses at least $\lceil \vartheta n/d \rceil$ packing patterns; in particular $\nu(I_n^{\vartheta}) \geq \lceil \vartheta n/d \rceil \to +\infty$ as $\vartheta \to +\infty$. Consequently, the asymptotic worst-case performance ratios of NF, FF, BF, WF and of their decreasing-order variants are unbounded as well.

4. Approximation via class-wise optimization and merging

In this section, we show that a 2-approximation algorithm for the BPPS can be obtained by applying a BPP algorithm $\mathcal{A}$ *class by class*, followed by a *merging phase* of compatible packing patterns. The analysis relies on the notion of pairwise merge-maximality, which is defined below. We first introduce the associated class-wise BPP instances.

For each class $c \in \mathcal{C}$, define the associated BPP instance I_c with item set $\mathcal{I}_c$, item weights $\{w_i\}_{i\in\mathcal{I}_c}$, and bin capacity $d_c := d - s_c$. By the standing assumption $w_i + s_c \leq d$, every item of class c fits into a bin of capacity d_c; hence, $d_c \geq 1$ and I_c is a feasible BPP instance. A feasible BPP solution for I_c is a family of nonempty subsets $\mathcal{S} \subseteq 2^{\mathcal{I}_c}$ forming a partition of $\mathcal{I}_c$ (i.e., $\bigcup_{S\in\mathcal{S}} S = \mathcal{I}_c$, $S \cap S' = \emptyset$ for all distinct $S, S' \in \mathcal{S}$); feasibility requires $\sum_{i\in S} w_i \leq d_c$ for all $S \in \mathcal{S}$. Given a BPP algorithm $\mathcal{A}$, we denote by $\mathcal{S}(\mathcal{A}, I_c)$ the solution returned by $\mathcal{A}$ on instance I_c, and by $\overline{\beta}_c := |\mathcal{S}(\mathcal{A}, I_c)|$ its number of open bins. We also denote by $\mathcal{S}^{\star}(I_c)$ an optimal solution to I_c, and by $\beta_c := |\mathcal{S}^{\star}(I_c)|$ the number of bins used by the solution.

The following definition formalizes a property, common to many classical BPP approximation algorithms, that is crucial to the theoretical analysis of the proposed algorithm: it characterizes the class-wise packings constructed in Phase 1 of our algorithms and supports all the approximation guarantees established below.

Definition 4.1. *Let I be a BPP instance with bin capacity $\hat{d}$, and let $\mathcal{S}$ be a feasible solution to I. We say that $\mathcal{S}$ is* pairwise merge-maximal *if no two of its packing patterns can be feasibly merged, that is, if*

$$\sum_{i\in S} w_i + \sum_{i\in S'} w_i > \hat{d} \qquad \text{for all distinct } S, S' \in \mathcal{S}. \tag{3}$$

Throughout the rest of the paper, we require $\mathcal{A}$ to return a pairwise merge-maximal solution to every BPP instance to which it is applied. In particular, when the input is I_c, the packing $\mathcal{S}(\mathcal{A}, I_c)$ returned by $\mathcal{A}$ satisfies condition (3) with $\hat{d} = d - s_c$.

A natural choice for $\mathcal{A}$ is an Any Fit algorithm. To see that every packing returned by such an algorithm is pairwise merge-maximal, consider two of its packing patterns S_1 and S_2, with S_1 opened before S_2, and let i be the first item assigned to S_2. When S_2 is opened, item i does not fit in S_1; hence, the load of S_1 at that time plus w_i exceeds the bin capacity. Since the load of S_1 can only increase thereafter and $i \in S_2$, the final loads of S_1 and S_2 also sum to more than the bin capacity. Thus, S_1 and S_2 cannot be merged. As the choice of the two patterns is arbitrary, every packing returned by an Any Fit algorithm is pairwise merge-maximal. The same property holds for every optimal BPP

packing: if two of its patterns could be feasibly merged, doing so would produce a feasible solution using one fewer bin, contradicting optimality. Finally, pairwise merge-maximality can be enforced on the output of any BPP algorithm by repeatedly merging feasible pairs of patterns until no such pair remains. Since merging never increases the number of open bins, this postprocessing preserves any approximation guarantee enjoyed by the original algorithm, and it runs in polynomial time.

We now introduce our two-phase algorithm for the BPPS, denoted by $\mathrm{TP}_{\mathcal{A}}$ and based on a BPP algorithm $\mathcal{A}$. As summarized in Algorithm 1, Phase 1 solves the BPP instance associated with each class independently and collects the resulting packing patterns. Phase 2 then repeatedly merges compatible patterns until no further merge is possible.

Algorithm 1: The two-phase algorithm $\mathrm{TP}_{\mathcal{A}}$

Input: A BPPS instance $I = (\mathcal{I}, C, d, r, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{f})$ and a BPP algorithm $\mathcal{A}$.
Output: A feasible BPPS solution $\mathcal{S}$.

`// Phase 1: class-wise packing`
$\mathcal{S} \leftarrow \emptyset$;
foreach *class* $c \in C$ **do**
  construct the BPP instance I_c with items $\mathcal{I}_c$, weights $\{w_i\}_{i \in \mathcal{I}_c}$, and capacity $d - s_c$;
  apply $\mathcal{A}$ to I_c and let $\mathcal{S}(\mathcal{A}, I_c)$ be the packing it returns;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}(\mathcal{A}, I_c)$;
end
$\mathcal{S}_1 \leftarrow \mathcal{S}$;
`// Phase 2: merge-maximal merging`
while *there are distinct* $S, S' \in \mathcal{S}$ *such that* $\ell(S \cup S') \leq d$ **do**
  choose any such pair (S, S');
  $\mathcal{S} \leftarrow \big(\mathcal{S} \setminus \{S, S'\}\big) \cup \{S \cup S'\}$;
end
$\mathcal{S}_2 \leftarrow \mathcal{S}$;
return $\mathcal{S}_2$;

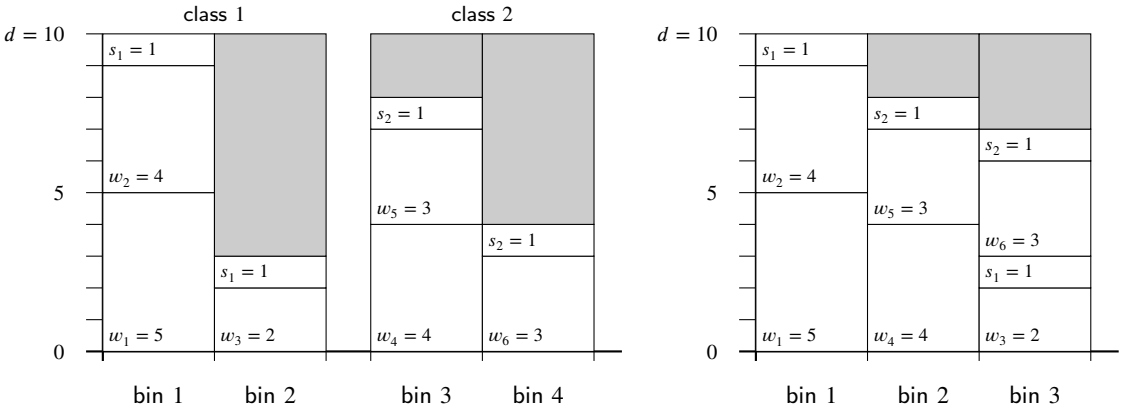
Following Algorithm 1, we denote by $\mathcal{S}_1$ the value of $\mathcal{S}$ at the end of Phase 1 and by $\mathcal{S}_2$ the solution returned at the end of Phase 2. The choice of the pair to merge in Phase 2 is left unspecified: all the theoretical properties proved below hold for any implementation that iterates the merging phase until no pair of packing patterns can be merged. We only require the selection rule not to depend on the bin cost r nor on the setup costs $\boldsymbol{f}$, so that $\mathrm{TP}_{\mathcal{A}}$ is cost-oblivious in the sense of Remark 2.1.

If $\mathcal{A}$ runs in polynomial time, so does $\mathrm{TP}_{\mathcal{A}}$. Indeed, Phase 1 runs $\mathcal{A}$ once per class. Since every Phase-1 pattern is nonempty, $|\mathcal{S}_1| \leq n$. Phase 2 hence performs at most $n - 1$ merges. At each iteration, all pairs of current patterns can be scanned, and the feasibility of each pair can be tested in polynomial time from their item loads and active-class sets. Thus Phase 2 runs in polynomial time as well.

Both $\mathcal{S}_1$ and $\mathcal{S}_2$ are feasible BPPS solutions. In Phase 1, for each class $c \in C$, $\mathrm{TP}_{\mathcal{A}}$ applies $\mathcal{A}$ to the BPP instance I_c with bin capacity $d - s_c$. Since $\mathcal{S}(\mathcal{A}, I_c)$ is a feasible BPP solution for I_c, its sets form a partition of $\mathcal{I}_c$. Because $\{\mathcal{I}_c\}_{c \in C}$ is a partition of $\mathcal{I}$, we have that $\mathcal{S}_1$ forms a partition of $\mathcal{I}$. Moreover, for any $S \in \mathcal{S}(\mathcal{A}, I_c)$, we have $\sum_{i \in S} w_i \leq d - s_c$; hence, $\ell(S) = \sum_{i \in S} w_i + s_c \leq d$. Therefore $\mathcal{S}_1$ is a feasible BPPS solution. Phase 2 starts from $\mathcal{S}_1$ and repeatedly replaces two packing patterns S and S' by $S \cup S'$ only when $\ell(S \cup S') \leq d$; therefore, by construction, $\mathcal{S}_2$ is a feasible BPPS solution. Figure 3 illustrates the solutions returned by $\mathrm{TP}_{\mathcal{A}}$ at the end of the two phases on a small BPPS instance when $\mathcal{A} = \mathrm{FF}$.

We now introduce the notation needed to establish the approximation ratio of $\mathrm{TP}_{\mathcal{A}}$. Consider an arbitrary feasible BPPS instance I, and let $\mathcal{S}^\star$ be an optimal solution to it, so that $UB(\mathcal{S}^\star) = OPT(I)$. For each class $c \in C$, let

$$q_c^\star := \Big|\big\{S \in \mathcal{S}^\star : c \in C(S)\big\}\Big|$$



(a) The Phase 1 solution $\mathcal{S}_1$

(b) The Phase 2 solution $\mathcal{S}_2$

Figure 3: Solution returned by $\mathrm{TP}_{\mathrm{FF}}$ at the end of the two phases on a BPPS instance with bin capacity $d = 10$, number of items $n = 6$, number of classes $m = 2$ and bin cost $r = 10$. The item weights are $w_1 = 5$, $w_2 = 4$, $w_3 = 2$, $w_4 = 4$, $w_5 = 3$ and $w_6 = 3$, 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 $\mathcal{I}_1 = \{1, 2, 3\}$ and $\mathcal{I}_2 = \{4, 5, 6\}$. The 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 items packed into it and the setup weight of its active classes. Shaded areas indicate unused capacity. In Phase 1 (left), FF is run separately on the two classes and opens four bins, yielding a total cost of $UB(\mathcal{S}_1) = 50$. In Phase 2 (right), the packing patterns containing items 3 and 6 are merged. No two of the resulting packing patterns can be further merged, so this is the final solution returned by $\mathrm{TP}_{\mathrm{FF}}$, with a total cost of $UB(\mathcal{S}_2) = 40$.

be the number of packing patterns of $\mathcal{S}^\star$ in which class c is active. Moreover, for any subset of classes $\tilde{\mathcal{C}} \subseteq \mathcal{C}$, define the load contributed by those classes in $\mathcal{S}^\star$ as

$$\ell^\star(\tilde{\mathcal{C}}) := \sum_{c \in \tilde{\mathcal{C}}} \left(\sum_{i \in \mathcal{I}_c} w_i + q_c^\star s_c \right). \tag{4}$$

Thus, $\ell^\star(\tilde{\mathcal{C}})$ accounts for the weight of all items belonging to classes in $\tilde{\mathcal{C}}$, together with the setup weight of each such class for every packing pattern in which it is active in $\mathcal{S}^\star$. In particular,

$$\ell^\star(\mathcal{C}) = \sum_{S \in \mathcal{S}^\star} \ell(S) \leq d\,|\mathcal{S}^\star|. \tag{5}$$

Finally, we call a class *small* if all its items fit into a single bin and *large* otherwise. Accordingly, we partition the set of classes into the set $\mathcal{C}_{\mathrm{S}}$ of small classes and the set $\mathcal{C}_{\mathrm{L}}$ of large classes, where

$$\mathcal{C}_{\mathrm{S}} := \left\{ c \in \mathcal{C} : \sum_{i \in \mathcal{I}_c} w_i + s_c \leq d \right\} \qquad \text{and} \qquad \mathcal{C}_{\mathrm{L}} := \left\{ c \in \mathcal{C} : \sum_{i \in \mathcal{I}_c} w_i + s_c > d \right\}.$$

The following lemma, which will be used to prove the approximation ratio of $\mathrm{TP}_{\mathcal{A}}$, shows that the number of Phase 1 bins used for each large class is less than twice the load contributed by that class in an optimal solution, divided by the bin capacity. This bound follows directly from pairwise merge-maximality of the packing returned by $\mathcal{A}$.

Lemma 4.2. *Let I be a feasible BPPS instance, let $\mathcal{S}^\star$ be an optimal solution to I and, for each class $c \in \mathcal{C}$, let $q_c^\star$ be the number of packing patterns of $\mathcal{S}^\star$ in which c is active. Let $c \in \mathcal{C}_{\mathrm{L}}$ and let $\mathcal{S}(\mathcal{A}, I_c)$ be a pairwise merge-maximal solution to I_c. Then,*

$$\overline{\beta}_c < \frac{2}{d} \left(\sum_{i \in \mathcal{I}_c} w_i + q_c^\star s_c \right). \tag{6}$$

Proof. Since c is a large class, its items do not fit in one bin of capacity $d - s_c$; hence $\overline{\beta}_c \geq 2$. Let $\mathcal{S}(\mathcal{A}, I_c) = \{S_1, \ldots, S_{\overline{\beta}_c}\}$. Since $\mathcal{S}(\mathcal{A}, I_c)$ is pairwise merge-maximal, by (3) we have

$$\sum_{i \in S_j} w_i + \sum_{i \in S_h} w_i > d - s_c \qquad \text{for all } 1 \leq j < h \leq \overline{\beta}_c.$$

Summing these inequalities over all $\binom{\overline{\beta}_c}{2}$ pairs (S_j, S_h), and observing that each S_j occurs in exactly $\overline{\beta}_c - 1$ such pairs, yields

$$(\overline{\beta}_c - 1) \sum_{i \in I_c} w_i > \binom{\overline{\beta}_c}{2} (d - s_c),$$

78 and hence $\sum_{i \in I_c} w_i > \frac{\overline{\beta}_c}{2}(d - s_c)$. Moreover, the $q_c^\star$ packing patterns of $\mathcal{S}^\star$ in which c is active induce a feasible packing for I_c, so $q_c^\star(d - s_c) \geq \sum_{i \in I_c} w_i$ and therefore $q_c^\star > \frac{\overline{\beta}_c}{2}$. Consequently,

$$\begin{aligned} 2\left(\sum_{i \in I_c} w_i + q_c^\star s_c\right) &= 2 \sum_{i \in I_c} w_i + 2 q_c^\star s_c \\ &> \overline{\beta}_c (d - s_c) + \overline{\beta}_c s_c \\ &= \overline{\beta}_c d, \end{aligned}$$

which proves (6). □

For every small class $c \in \mathcal{C}_S$, pairwise merge-maximality also implies $\overline{\beta}_c = 1$. If a feasible packing opened at least two bins, the item loads of any two of its bins would sum to at most the total item weight of the class, which is at most $d - s_c$, contradicting (3).

Pairwise merge-maximality further implies that every merge performed in Phase 2 involves packing patterns with disjoint sets of active classes. To see this, suppose that two current packing patterns both contain a certain class $c \in \mathcal{C}$. Each current packing pattern is the union of some Phase 1 BPP patterns and therefore contains a distinct original pattern of $\mathcal{S}(\mathcal{A}, I_c)$. By (3), the combined item weight of class c in these two original patterns exceeds $d - s_c$. Their union includes those items and incurs setup weight s_c, so its load exceeds d, independently of its other contents. The proposed merge is therefore infeasible. It follows that every feasible Phase 2 merge combines disjoint active-class sets and that the number of activations of each class remains equal to $\overline{\beta}_c$ after Phase 2.

We are now ready to state the main approximation result for $\mathrm{TP}_{\mathcal{A}}$. The factor 2 follows from two related consequences of pairwise merge-maximality. For each large class, Lemma 4.2 bounds the number of Phase 1 bins in terms of the item and setup load contributed by that class in $\mathcal{S}^\star$. For the small classes, after Phase 2 the remaining packing patterns containing only small classes are pairwise non-mergeable; consequently, if at least two such patterns remain, their average load is greater than $d/2$. These two bounds control the bin-opening cost, while the same class-wise analysis show that no class is activated in the $\mathrm{TP}_{\mathcal{A}}$ solution more than twice as often as in $\mathcal{S}^\star$, controlling the setup costs.

Theorem 4.3. *Let $\mathcal{A}$ be a BPP algorithm that returns a pairwise merge-maximal solution to every input instance. Then, for every feasible BPPS instance I, the solution $\mathcal{S}_2$ returned by* $\mathrm{TP}_{\mathcal{A}}$ *satisfies*

$$UB(\mathcal{S}_2) \leq 2\, OPT(I).$$

Proof. Consider an arbitrary feasible BPPS instance I and an optimal solution $\mathcal{S}^\star$ to it. We first bound the number of packing patterns in $\mathcal{S}_2$, by proving that $|\mathcal{S}_2| \leq 2|\mathcal{S}^\star|$. By Lemma 4.2,

$$\sum_{c \in \mathcal{C}_L} \overline{\beta}_c \leq \frac{2}{d} \ell^\star(\mathcal{C}_L), \tag{7}$$

and the inequality is strict whenever $\mathcal{C}_L \neq \emptyset$. Moreover,

$$\ell^\star(\mathcal{C}_L) + \ell^\star(\mathcal{C}_S) = \ell^\star(\mathcal{C}) \leq d\, |\mathcal{S}^\star|, \tag{8}$$

where the inequality follows from (5).

Let us now consider the partition of $\mathcal{S}_2$ into

$$\mathcal{F}_{\mathrm{L}} := \{S \in \mathcal{S}_2 : \mathcal{C}(S) \cap \mathcal{C}_{\mathrm{L}} \neq \emptyset\}$$

and

$$\mathcal{F}_{\mathrm{S}} := \mathcal{S}_2 \setminus \mathcal{F}_{\mathrm{L}}.$$

Every packing pattern in $\mathcal{F}_{\mathrm{L}}$ contains at least one of the $\sum_{c \in \mathcal{C}_{\mathrm{L}}} \overline{\beta}_c$ Phase 1 packing patterns dedicated to large classes. Since Phase 2 only merges existing packing patterns,

$$|\mathcal{F}_{\mathrm{L}}| \leq \sum_{c \in \mathcal{C}_{\mathrm{L}}} \overline{\beta}_c. \tag{9}$$

On the other hand, every packing pattern in $\mathcal{F}_{\mathrm{S}}$ contains only small classes. Moreover, each small class corresponds to exactly one Phase 1 packing pattern and thus occurs in exactly one Phase 2 packing pattern. Hence, the active-class sets of the elements of $\mathcal{F}_{\mathrm{S}}$ are pairwise disjoint subsets of $\mathcal{C}_{\mathrm{S}}$. Consequently,

$$\begin{aligned}\sum_{S \in \mathcal{F}_{\mathrm{S}}} \ell(S) &\leq \sum_{c \in \mathcal{C}_{\mathrm{S}}} \left(\sum_{i \in \mathcal{I}_c} w_i + s_c \right) \\ &\leq \sum_{c \in \mathcal{C}_{\mathrm{S}}} \left(\sum_{i \in \mathcal{I}_c} w_i + q_c^\star s_c \right) \\ &= \ell^\star(\mathcal{C}_{\mathrm{S}}),\end{aligned} \tag{10}$$

where the second inequality follows from $q_c^\star \geq 1$ for all $c \in \mathcal{C}$.

We distinguish two cases. Suppose first that $|\mathcal{F}_{\mathrm{S}}| \geq 2$. At the end of Phase 2, no two distinct packing patterns $S, S' \in \mathcal{F}_{\mathrm{S}}$ can be merged. Moreover, their active class sets are disjoint, because every small class occurs in only one Phase 2 packing pattern. Thus,

$$\ell(S) + \ell(S') = \ell(S \cup S') > d \qquad \text{for all distinct } S, S' \in \mathcal{F}_{\mathrm{S}}.$$

Summing these inequalities as in the proof of Lemma 4.2 over all pairs and observing that each packing pattern appears in exactly $|\mathcal{F}_{\mathrm{S}}| - 1$ pairs yields $\sum_{S \in \mathcal{F}_{\mathrm{S}}} \ell(S) > \frac{d}{2} |\mathcal{F}_{\mathrm{S}}|$. Together with (10), this gives

$$|\mathcal{F}_{\mathrm{S}}| < \frac{2}{d} \ell^\star(\mathcal{C}_{\mathrm{S}}). \tag{11}$$

Combining (7), (9), and (11), we obtain

$$\begin{aligned}|\mathcal{S}_2| &= |\mathcal{F}_{\mathrm{L}}| + |\mathcal{F}_{\mathrm{S}}| \\ &\leq \sum_{c \in \mathcal{C}_{\mathrm{L}}} \overline{\beta}_c + |\mathcal{F}_{\mathrm{S}}| \\ &< \frac{2}{d} \left(\ell^\star(\mathcal{C}_{\mathrm{L}}) + \ell^\star(\mathcal{C}_{\mathrm{S}}) \right) \\ &\leq 2|\mathcal{S}^\star|.\end{aligned}$$

This proves the desired bound in the first case.

It remains to consider the case $|\mathcal{F}_{\mathrm{S}}| \leq 1$. If $\mathcal{C}_{\mathrm{L}} = \emptyset$, then $|\mathcal{F}_{\mathrm{L}}| = 0$ and $|\mathcal{S}_2| = |\mathcal{F}_{\mathrm{S}}| \leq 1$, so the desired bound is immediate. Otherwise, $\mathcal{C}_{\mathrm{L}} \neq \emptyset$ and the strict version of (7) gives

$$|\mathcal{S}_2| = |\mathcal{F}_{\mathrm{L}}| + |\mathcal{F}_{\mathrm{S}}|$$

$$
\begin{aligned}
&\leq \sum_{c \in C_{\mathrm{L}}} \overline{\beta}_c + 1 \\
&< \frac{2}{d} \ell^\star(C_{\mathrm{L}}) + 1 \\
&\leq \frac{2}{d} \left(\ell^\star(C_{\mathrm{L}}) + \ell^\star(C_{\mathrm{S}}) \right) + 1 \\
&\leq 2|\mathcal{S}^\star| + 1.
\end{aligned}
$$

Since both $|\mathcal{S}_2|$ and $2|\mathcal{S}^\star|$ are integers, the strict inequality implies the desired bound in this case as well. Thus, in all cases,

$$|\mathcal{S}_2| \leq 2|\mathcal{S}^\star|. \tag{12}$$

We finally bound the setup-cost contribution. Phase 2 only merges packing patterns with disjoint active class sets, and therefore the number of activations of every class in $\mathcal{S}_2$ does not change with respect to the solution obtained after Phase 1. In other words, each class $c \in C$ is active exactly in $\overline{\beta}_c$ packing patterns in $\mathcal{S}_2$. For $c \in C_{\mathrm{S}}$, $\overline{\beta}_c = 1 \leq q_c^\star$. For $c \in C_{\mathrm{L}}$, by Lemma 4.2 we have that $\overline{\beta}_c < 2q_c^\star$. Consequently, since all setup costs are nonnegative,

$$\sum_{S \in \mathcal{S}_2} \sum_{c \in C(S)} f_c = \sum_{c \in C} \overline{\beta}_c f_c \leq 2 \sum_{c \in C} q_c^\star f_c = 2 \sum_{S \in \mathcal{S}^\star} \sum_{c \in C(S)} f_c. \tag{13}$$

Combining (12) and (13), we conclude that

$$
\begin{aligned}
UB(\mathcal{S}_2) &= r\,|\mathcal{S}_2| + \sum_{S \in \mathcal{S}_2} \sum_{c \in C(S)} f_c \\
&\leq 2r\,|\mathcal{S}^\star| + 2 \sum_{S \in \mathcal{S}^\star} \sum_{c \in C(S)} f_c \\
&= 2\,UB(\mathcal{S}^\star) = 2\,OPT(I).
\end{aligned}
$$

□

Corollary 4.4. *If $\mathcal{A}$ is a polynomial-time algorithm for the BPP that returns a pairwise merge-maximal solution to every input instance,* $\mathrm{TP}_{\mathcal{A}}$ *is a 2-approximation algorithm for the BPPS.*

The factor 2 in Theorem 4.3 is tight. The following construction contains only small classes and does not depend on the Phase 2 merge-selection rule, since the load of every Phase 1 packing pattern is already too large to allow any feasible merge. Hence, this guarantee holds for every possible merge-selection rule in Phase 2, provided that merging continues until no pair of packing patterns can be feasibly merged.

Proposition 4.5. *For every BPP algorithm $\mathcal{A}$ that returns pairwise merge-maximal solutions, the absolute worst-case performance ratio of* $\mathrm{TP}_{\mathcal{A}}$ *is*

$$R_{\mathrm{TP}_{\mathcal{A}}} = 2.$$

Proof. For any integer $q \geq 2$, consider a BPPS instance I_q with bin capacity $d = 2q(q+1)$, bin cost $r = 1$, and $2q$ classes, each containing $t = q(q+1)$ items. Index the items class by class, so that class $c \in C$ contains the items with indices in $\{(c-1)t + 1, \ldots, ct\}$. Thus, there are $2qt = 2q^2(q+1)$ items in total. Every item $i \in \mathcal{I}$ has weight $w_i = 1$, while every class $c \in C$ has setup weight $s_c = 1$ and setup cost $f_c = 0$. Every class is small because

$$\sum_{i \in \mathcal{I}_c} w_i + s_c = q(q+1) + 1 \leq 2q(q+1) = d.$$

The total item weight of a class is $q(q+1) = d/2$, so all its items fit in one bin of the associated BPP instance, whose capacity is $d - s_c = d - 1$. Hence, $\mathcal{A}$ returns exactly one bin for each class (by pairwise merge-maximality of the

class-wise BPP solutions). Phase 1 consequently produces $2q$ packing patterns, each of load $\frac{d}{2}+1$. No two of them can be merged in Phase 2, since their union has load

$$2\left(\frac{d}{2}+1\right) = d+2 > d.$$

Thus $\mathrm{TP}_{\mathcal{A}}(I_q) = 2q$, independently of the merge-selection rule.

Notice that the total item weight is

$$2q \cdot q(q+1) = 2q^2(q+1) = qd.$$

Thus, q bins would be completely filled by the items alone and could not accommodate any of the setup weights, which are strictly positive. Every feasible solution to I_q therefore contains at least $q+1$ packing patterns, each of which must be assigned to a bin. This lower bound is attained by opening $q+1$ bins and, for every $b \in \{1,2,\dots,q+1\}$ and $c \in \{1,2,\dots,2q\}$, assigning to bin b the q class-c items with indices from $(c-1)t+(b-1)q+1$ through $(c-1)t+bq$. All the $t = q(q+1)$ items of each class are thereby assigned across the $q+1$ packing patterns, each having item load $2q^2$ and setup load $2q$. Hence every corresponding bin is full, because

$$2q^2 + 2q = 2q(q+1) = d.$$

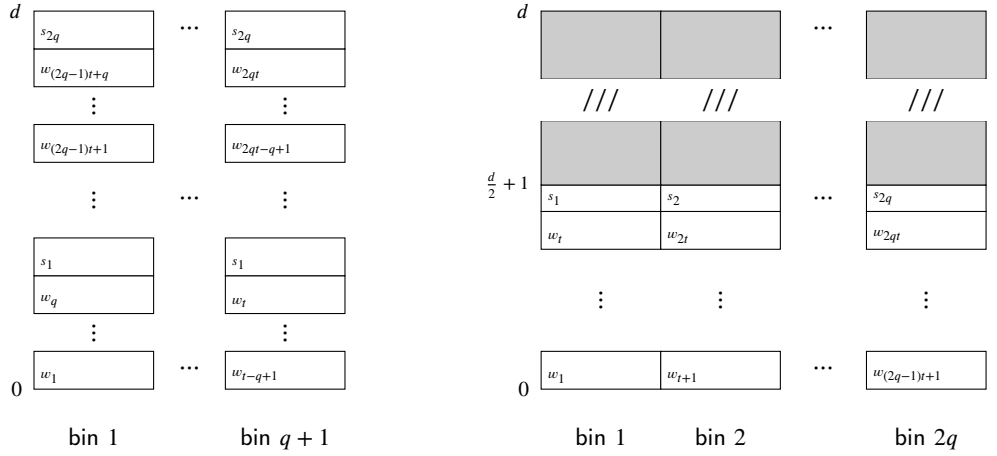
Consequently,

$$OPT(I_q) = q+1 \qquad \text{and} \qquad \frac{\mathrm{TP}_{\mathcal{A}}(I_q)}{OPT(I_q)} = \frac{2q}{q+1} \xrightarrow[q\to\infty]{} 2.$$

Since the absolute worst-case performance ratio is the supremum over all instances, this family gives $R_{\mathrm{TP}_{\mathcal{A}}} \geq 2$. The reverse inequality follows from Theorem 4.3 and, therefore, $R_{\mathrm{TP}_{\mathcal{A}}} = 2$. □

Corollary 4.6. *For every exact BPP algorithm $\mathcal{A}$, $R_{\mathrm{TP}_{\mathcal{A}}} = 2$.*

Figure 4 compares an optimal solution and the solution returned by $\mathrm{TP}_{\mathcal{A}}$ for the family of instances described in the proof of Proposition 4.5.



(a) An optimal BPPS solution

(b) The $\mathrm{TP}_{\mathcal{A}}$ solution

Figure 4: Worst-case family of instances for $\mathrm{TP}_{\mathcal{A}}$ used in the proof of Proposition 4.5. The items are ordered as specified in the proof. Left: an optimal BPPS solution consists of $q+1$ packing patterns, each containing q items from every class. Right: $\mathrm{TP}_{\mathcal{A}}$ produces one packing pattern for each of the $2q$ classes; each pattern contains all $q(q+1)$ items of one class and has $d/2-1$ units of unused capacity.

This construction demonstrates that the class-wise decomposition is the main source of loss in $\mathrm{TP}_{\mathcal{A}}$. Every associated BPP instance is solved to optimality—indeed, all items of a class fit in a single bin—but the Phase 1 packing patterns

have load greater than $d/2$ and Phase 2 cannot merge any pair of them. The global optimum instead splits every class across several packing patterns and exploits the resulting cross-class combinations. Thus, as stated in Corollary 4.6, even using an exact Phase 1 algorithm can produce solutions whose value is arbitrarily close to twice the optimum. Improving the general factor 2 therefore requires reconsidering assignments made in Phase 1, rather than just solving the class-wise subproblems more accurately or changing the merge order of Phase 2.

Theorem 4.3 derives the same factor for the bin-opening cost and the setup costs. The factor of the latter, however, can be refined whenever the BPP algorithm used in Phase 1, besides returning pairwise merge-maximal solutions, is an α-approximation algorithm with $\alpha \leq 2$; the refinement is potentially strict when $\alpha < 2$—as is the case for FF, BF, FFD and BFD—and it is strict whenever, in addition, the setup-cost component of an optimal solution is positive. Indeed, the packing patterns of an optimal BPPS solution in which class $c \in C$ is active induce a feasible solution for I_c, featuring $q_c^\star$ open bins. Hence $\beta_c \leq q_c^\star$, and an α-approximation algorithm opens at most $\alpha q_c^\star$ bins for that class. The factor 2 remains necessary for the total number of open bins, but the setup costs can be charged with factor α. This is formalized in the following corollary.

Corollary 4.7. *Let $\mathcal{A}$ be an α-approximation algorithm for the BPP with $\alpha \leq 2$ and which returns a pairwise merge-maximal solution to every input BPP instance. Let I be a feasible BPPS instance, let $S^\star$ be an optimal solution to I and, for each class $c \in C$, let $q_c^\star$ be the number of packing patterns of $S^\star$ in which c is active. Then, the objective value of the solution returned by $\mathrm{TP}_\mathcal{A}$ on I satisfies*

$$UB(S_2) \leq 2r|S^\star| + \alpha \sum_{c \in C} q_c^\star f_c = 2\,OPT(I) - (2-\alpha) \sum_{c \in C} q_c^\star f_c.$$

Proof. The bound (12) gives

$$r|S_2| \leq 2r|S^\star|.$$

For each class $c \in C$, the $q_c^\star$ packing patterns of $S^\star$ in which c is active induce a feasible packing of I_c, and therefore $\beta_c \leq q_c^\star$. Since $\mathcal{A}$ is an α-approximation algorithm for the BPP, it follows that

$$\overline{\beta}_c \leq \alpha \beta_c \leq \alpha q_c^\star.$$

As observed in the proof of Theorem 4.3, Phase 2 does not increase the number of activations of any class. It follows that

$$\sum_{S \in S_2} \sum_{c \in C(S)} f_c = \sum_{c \in C} \overline{\beta}_c f_c \leq \alpha \sum_{c \in C} q_c^\star f_c.$$

Adding the two bounds proves the first inequality. The equality follows from $OPT(I) = r|S^\star| + \sum_{c \in C} q_c^\star f_c$. □

Note that the refinement vanishes on instances with zero setup costs, which is precisely the case of the tight family of Proposition 4.5. Equivalently, letting $\sigma(I) := \sum_{c \in C} q_c^\star f_c / OPT(I) \in [0, 1)$ denote the share of the cost of $S^\star$ due to class activations, Corollary 4.7 reads

$$\frac{UB(S_2)}{OPT(I)} \leq 2 - (2-\alpha)\,\sigma(I),$$

indicating that the loss with respect to the optimum decreases as the setup costs become more relevant.

Theorem 4.3 and Proposition 4.5 apply, in particular, to the standard Any Fit algorithms discussed in Section 2. Sorting the items within each class does not affect pairwise merge-maximality: FFD, BFD, and WFD still open a new bin only when the current item fits in none of the previously opened bins. Thus, FF, BF, and WF, as well as their decreasing-order variants, give the following immediate consequence.

Corollary 4.8. *The absolute worst-case performance ratios of* $\mathrm{TP}_{\mathrm{FF}}$, $\mathrm{TP}_{\mathrm{BF}}$, $\mathrm{TP}_{\mathrm{WF}}$, $\mathrm{TP}_{\mathrm{FFD}}$, $\mathrm{TP}_{\mathrm{BFD}}$, *and* $\mathrm{TP}_{\mathrm{WFD}}$ *are equal to* 2.

Finally, the guarantees established in this section hold for the asymptotic measure of Remark 3.4 as well. The bound of Theorem 4.3 holds for every instance and, in particular, for instances with arbitrarily many open bins, so that $R^\infty_{\mathrm{TP}_\mathcal{A}} \leq 2$. Conversely, the family of Proposition 4.5 satisfies $\nu(I_q) = q + 1 \to +\infty$ as q goes to $+\infty$, so the matching lower bound is asymptotic in nature. Consequently $R^\infty_{\mathrm{TP}_\mathcal{A}} = R_{\mathrm{TP}_\mathcal{A}} = 2$, and the ratios of Corollary 4.8 are asymptotic worst-case performance ratios as well.

5. Conclusions

We studied approximation algorithms for the BPPS. First, we considered direct extensions of classical BPP algorithms to this variant. Although NF, FF, BF, and WF have finite worst-case performance ratios for the BPP, we proved by counterexample that their straightforward BPPS adaptations, as well as those of their decreasing-order variants, have unbounded absolute worst-case performance ratios. The counterexamples have unit-weight items and zero setup costs, showing that setup weights alone suffice to invalidate the classical guarantees.

We then introduced the two-phase algorithm $\mathrm{TP}_{\mathcal{A}}$, which packs each class separately in Phase 1 and repeatedly merges compatible packing patterns in Phase 2. We prove that the solution returned by $\mathrm{TP}_{\mathcal{A}}$ has cost at most twice the optimum under the assumption that $\mathcal{A}$ returns pairwise merge-maximal solutions—this property holds for both Any Fit and exact BPP algorithms, and can be obtained by postprocessing any class-wise packing. Hence, if $\mathcal{A}$ also runs in polynomial time, $\mathrm{TP}_{\mathcal{A}}$ is a 2-approximation algorithm for the BPPS. The bound is tight, even when all classes fit in a single bin, setup weights are equal to one, setup costs are zero, and every class-wise BPP instance is solved to optimality. It follows that $\mathrm{TP}_{\mathrm{FF}}$, $\mathrm{TP}_{\mathrm{BF}}$, $\mathrm{TP}_{\mathrm{WF}}$, and their decreasing-order variants have an absolute worst-case performance ratio exactly 2. When $\mathcal{A}$ is an α-approximation algorithm for the BPP with $\alpha \leq 2$, returning pairwise merge-maximal solutions, we further show that the cost of the solution returned by $\mathrm{TP}_{\mathcal{A}}$ is bounded by 2 times the bin-opening cost of an optimal solution and by α times its setup-cost component. On the negative side, since the BPP is a special case of the BPPS, no polynomial-time algorithm for the latter can have an absolute worst-case performance ratio smaller than $3/2$ unless $\mathsf{P} = \mathsf{NP}$.

Future work includes, first and foremost, understanding whether the factor-2 approximation can be improved. By Proposition 2.2, the absolute approximability threshold of the BPPS lies between $3/2$ and 2, and closing this gap is, in our opinion, the main question left open by this work. The family of instances we used to prove the tightness of the absolute worst-case performance ratio of $\mathrm{TP}_{\mathcal{A}}$ shows that neither solving the class-wise BPP instances more accurately, even exactly, nor changing the merging rule of Phase 2 would improve the factor 2. Thus, achieving a better factor likely requires coordinating class activations and item assignments across classes, either within or after Phase 1. It would also be natural to investigate stronger guarantees under structural restrictions, such as a bounded number of classes, setup weights or costs being related to bin capacity or bin cost, or additional restrictions on item weights.

References

Baldacci, R., Ciccarelli, F., Coniglio, S., Dose, V., Furini, F., 2025. The Bin Packing Problem with Setups: Formulations, Structural Properties and Computational Insights. arXiv:2509.10075.

Baldacci, R., Coniglio, S., Cordeau, J.F., Furini, F., 2024. A numerically exact algorithm for the bin-packing problem. INFORMS Journal on Computing 36, 141–162.

Barkel, M., Delorme, M., Malaguti, E., Monaci, M., 2025. Bounds and heuristic algorithms for the bin packing problem with minimum color fragmentation. European Journal of Operational Research 320, 57–68.

Bergman, D., Cardonha, C., Mehrani, S., 2019. Binary decision diagrams for bin packing with minimum color fragmentation, in: Rousseau, L., Stergiou, K. (Eds.), CPAIOR 2019, Springer. pp. 57–66.

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

Boyar, J., Dósa, G., Epstein, L., 2012. On the absolute approximation ratio for first fit and related results. Discrete Applied Mathematics 160, 1914–1923.

Coffman, E.G.J., Csirik, J., Galambos, G., Martello, S., Vigo, D., 2013. Bin packing approximation algorithms: survey and classification, in: Handbook of combinatorial optimization. Springer, pp. 455–531.

da Silva, R.F.F., Schouery, R.C.S., 2025. Solving cutting stock problems via an extended Ryan-Foster branching scheme and fast column generation. INFORMS Journal on Computing (in press).

Dawande, M., Kalagnanam, J., Sethuraman, J., 2001. Variable sized bin packing with color constraints. Electronic Notes in Discrete Mathematics 7, 154–157.

Dósa, G., Li, R., Han, X., Tuza, Z., 2013. Tight absolute bound for first fit decreasing bin-packing: $\mathrm{FFD}(L) \leq 11/9\,\mathrm{OPT}(L) + 6/9$. Theoretical Computer Science 510, 13–61.

Dósa, G., Sgall, J., 2013. First fit bin packing: A tight analysis, in: Portier, N., Wilke, T. (Eds.), 30th Symposium on Theoretical Aspects of Computer Science (STACS 2013), Schloss Dagstuhl – Leibniz-Zentrum für Informatik. pp. 538–549.

Dósa, G., Sgall, J., 2014. Optimal analysis of best fit bin packing, in: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (Eds.), Automata, Languages, and Programming (ICALP 2014), Springer. pp. 429–441.

Epstein, L., 2021. On bin packing with clustering and bin packing with delays. Discrete Optimization 41, 100647.

Epstein, L., 2023. Several methods of analysis for cardinality constrained bin packing. Theoretical Computer Science 942, 213–229.

Epstein, L., Imreh, C., Levin, A., 2010. Class constrained bin packing revisited. Theoretical Computer Science 411, 3073–3089.

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

Galambos, G., Martello, S., Vigo, D., 2025. An updated overview of bin packing approximation algorithms, in: Pardalos, P.M., Du, D.Z., Thai, M.T. (Eds.), Handbook of Combinatorial Optimization. Springer New York, New York, NY, pp. 1–85.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York.

Johnson, D.S., 1973. Near-optimal bin packing algorithms. Ph.D. thesis. Massachusetts Institute of Technology.

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

Mehrani, S., Cardonha, C., Bergman, D., 2022. Models and algorithms for the bin-packing problem with minimum color fragmentation. INFORMS Journal on Computing 34, 1070–1085.

Michel, S., Perrot, N., Vanderbeck, F., 2009. Knapsack problems with setups. European Journal of Operational Research 196, 909–918.

Pferschy, U., Scatamacchia, R., 2018. Improved dynamic programming and approximation results for the knapsack problem with setups. International Transactions in Operational Research 25, 667–682.

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

Shachnai, H., Tamir, T., 2004. Tight bounds for online class-constrained packing. Theoretical Computer Science 321, 103–123.

Simchi-Levi, D., 1994. New worst-case results for the bin-packing problem. Naval Research Logistics 41, 579–585.

Twigg, A., Xavier, E.C., 2015. Locality-preserving allocations problems and coloured bin packing. Theoretical Computer Science 596, 12–22.

Wei, L., Luo, Z., Baldacci, R., Lim, A., 2020. A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. INFORMS Journal on Computing 32, 428–443.

Xavier, E., Miyazawa, F., 2008. The class constrained bin packing problem with applications to video-on-demand. Theoretical Computer Science 393, 240–259.

A. Notation summary

Table 1 summarizes the notation used throughout the paper.

Table 1: Summary of notation

BPP and BPPS input	**Description**
$d \in \mathbb{Z}_{\geq 1}$	Bin capacity
$r \in \mathbb{Z}_{\geq 1}$	Fixed cost of each open bin
$n \in \mathbb{Z}_{\geq 1}$	Number of items
$\mathcal{I} = \{1, 2, \dots, n\}$	Set of items
$w_i \in \mathbb{Z}_{\geq 1}$	Weight of item $i \in \mathcal{I}$
$m \in \mathbb{Z}_{\geq 1}$	Number of classes
$\mathcal{C} = \{1, 2, \dots, m\}$	Set of item classes
$\mathcal{I}_c \subseteq \mathcal{I}$	Set of items belonging to class $c \in \mathcal{C}$
$s_c \in \mathbb{Z}_{\geq 0}$	Setup weight of class $c \in \mathcal{C}$
$f_c \in \mathbb{Z}_{\geq 0}$	Setup cost of class $c \in \mathcal{C}$
BPPS solutions	**Description**
$\mathcal{C}(S)$	Set of classes active in a subset $S \subseteq \mathcal{I}$
$\ell(S)$	Load of S, including the weights of its items and of its active-class setups
$\mathcal{S} \subseteq 2^{\mathcal{I}}$	Feasible BPPS solution, i.e., a partition of $\mathcal{I}$ into nonempty packing patterns
$S \in \mathcal{S}$	Packing pattern assigned to one bin
$UB(\mathcal{S})$	Cost of a feasible BPPS solution $\mathcal{S}$
$OPT(I)$	Cost of an optimal solution to a BPPS instance I
$\mathcal{S}^\star$	An optimal solution to the BPPS instance under consideration
Class-wise BPP instances and two-phase heuristic	**Description**
I_c	BPP instance induced by class c, with item set $\mathcal{I}_c$ and bin capacity $d - s_c$
$d_c := d - s_c$	Bin capacity of the class-wise BPP instance I_c
$\mathcal{A}$	BPP algorithm used in Phase 1 of the two-phase heuristic
$\mathcal{S}(\mathcal{A}, I_c)$	Solution returned by a BPP algorithm $\mathcal{A}$ to I_c
$\overline{\beta}_c := \lvert\mathcal{S}(\mathcal{A}, I_c)\rvert$	Number of open bins in the class-wise solution $\mathcal{S}(\mathcal{A}, I_c)$
$\mathcal{S}^\star(I_c)$	Optimal solution to I_c
$\beta_c := \lvert\mathcal{S}^\star(I_c)\rvert$	Optimal number of bins for I_c
$\mathrm{TP}_{\mathcal{A}}$	Two-phase BPPS heuristic using algorithm $\mathcal{A}$ in Phase 1
$\mathcal{S}_1$	Solution obtained after the class-wise packing phase of $\mathrm{TP}_{\mathcal{A}}$
$\mathcal{S}_2$	Solution returned after the merging phase of $\mathrm{TP}_{\mathcal{A}}$
Approximation analysis	**Description**
I	Instance of the BPPS (of the BPP, where explicitly stated)
$\mathcal{A}(I)$	Value of the solution returned by algorithm $\mathcal{A}$ on instance I
$R_{\mathcal{A}}$	Absolute worst-case performance ratio of an approximation algorithm $\mathcal{A}$
$R_{\mathcal{A}}^{\infty}$	Asymptotic worst-case performance ratio of an approximation algorithm $\mathcal{A}$
$\nu(I)$	Minimum number of packing patterns over all optimal solutions to a BPPS instance I
α	Approximation ratio of an approximation algorithm $\mathcal{A}$
$q_c^\star$	Number of packing patterns of an optimal BPPS solution $\mathcal{S}^\star$ in which class c is active
$\ell^\star(\tilde{\mathcal{C}})$	Load contributed in an optimal BPPS solution $\mathcal{S}^\star$ by the classes in $\tilde{\mathcal{C}} \subseteq \mathcal{C}$
$\mathcal{C}_{\mathrm{S}}$	Set of small classes, i.e., those whose items fit in a single bin
$\mathcal{C}_{\mathrm{L}}$	Set of large classes, i.e., those whose items do not fit in a single bin
$\mathcal{F}_{\mathrm{S}}$	Packing patterns of $\mathcal{S}_2$ containing items of small classes only
$\mathcal{F}_{\mathrm{L}}$	Packing patterns of $\mathcal{S}_2$ containing items of at least one large class