A first approximation algorithm for the Bin Packing Problem with Setups

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 study constant-factor approximation algorithms for the *Bin Packing Problem with Setups* (BPPS). First, we show that adaptations of classical BPP heuristics can have arbitrarily poor worst-case performance on BPPS instances. Then, we propose a two-phase heuristic for the BPPS that applies an α-approximation algorithm for the BPP to the items of each class and then performs a merging phase on the open bins. We prove that this heuristic is a $2\,\alpha$-approximation algorithm for the BPPS.

Keywords: Combinatorial Optimization, Bin Packing Problem, Approximation algorithms

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 $\mathcal{I} = \{1, 2, \ldots, n\}$ 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 *classes*. For each class $c \in \mathcal{C} = \{1, 2, \ldots, m\}$, 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.

The *Bin Packing Problem with Setups* (BPPS) asks for a minimum-cost partition of the items into bins such that, in each bin, the sum of the weights of the assigned items and the setup weights of the classes active in the bin does not exceed the bin capacity. The cost of a solution is obtained by summing, over all open bins, the bin cost plus the setup costs of the classes active in the bin. We denote by $\psi(I)$ the optimal value of an instance I of the BPPS, i.e., the minimum total cost required to pack all items. To ensure the feasibility of a BPPS instance, we assume that, for each class $c \in \mathcal{C}$,

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

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 in which all items can be packed into a single bin.

The BPPS is strongly $\mathcal{NP}$-hard, as it admits the classical *Bin Packing Problem* (BPP) as a special case. The BPPS was recently introduced in [1], which, to the best of our knowledge, remains the only study devoted to this important problem, with applications in production planning and logistics. The authors of [1] propose a natural integer linear programming (ILP) formulation, analyze its linear programming relaxation, and derive strengthening valid inequalities which guarantee a worst-case performance ratio of 1/2 for the resulting lower bound with respect to the optimal BPPS value. The computational experiments in [1] on a large set of benchmark instances show that the proposed enhancements substantially improve the performance of the ILP model.

To the best of our knowledge, no approximation algorithm has been proposed so far 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 [2, 4]—it is natural to ask whether the BPPS inherits some of its favorable approximability properties. In this letter, we take a first exploratory step in this direction and start investigating this question.

2. Classical BPP heuristics

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 $z(I)$ 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 $\mathcal{A}(I) \geq z(I)$ and $\mathcal{A}(I)/z(I) \geq 1$. If there exists a finite constant $\alpha \in [1, +\infty)$ such that

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

then $\mathcal{A}$ is called an *α-approximation algorithm*. Moreover, following the notation used in, e.g., [3], the *worst-case performance ratio* of $\mathcal{A}$ is the smallest value of α for which $\mathcal{A}$ is an α-approximation algorithm. If no such finite α exists, the ratio is *unbounded* and $\mathcal{A}$ does not provide any worst-case guarantee.

We now briefly review classical *online heuristics* for the BPP, i.e., algorithms that process the items according to their given order in the instance. *Next Fit* (NF) maintains exactly one open bin and packs the current item into it if it fits; otherwise, it closes the bin and opens a new one. *First Fit* (FF) packs the current item into the lowest-indexed open bin into which it fits, opening a new bin if necessary. *Best Fit* (BF) packs the current item into a feasible open bin that leaves the minimum residual capacity (equivalently, among feasible bins, it chooses one of maximum load), opening a new bin if necessary. These online heuristics 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. These heuristics are called *Next Fit Decreasing* (NFD), *First Fit Decreasing* (FFD), and *Best Fit Decreasing* (BFD).

All the heuristics mentioned above are constant-factor approximation algorithms for the BPP. We refer the reader to the surveys [2, 4] for further details and for an overview of other constant-factor approximation algorithms for the BPP. It is worth mentioning that, unless $\mathcal{P} = \mathcal{NP}$, a worst-case performance ratio of $3/2$ is best possible for the BPP by any polynomial-time algorithm; moreover, this guarantee is achieved by FFD and BFD, as shown in [5].

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

In this section, we show that straightforward adaptations of the classical BPP heuristics NF, FF, and BF have no worst-case guarantee on BPPS instances. The same holds for their decreasing-order variants NFD, FFD, and BFD.

For the BPPS, these algorithms process items in the given order and apply their usual packing rule (next, first feasible, or best feasible), where feasibility and residual capacity account for class setup weights. In particular, when an item is packed into a bin in which its 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.

We start with NF and exhibit a family of BPPS instances on which NF opens one bin per item, whereas an optimal BPPS solution packs all items into only two bins.

Proposition 1. *For the* BPPS*, the worst-case performance ratio of* NF *is unbounded.*

Proof. Fix an even n 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 $c \in \{1, 2\}$,

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

So, packing each class in a separate bin is feasible. Moreover, no bin can contain items from both classes, since $s_1 + s_2 + 2 = n > d$. Hence $\psi(I_n) = 2\,r$, where r is the bin cost.

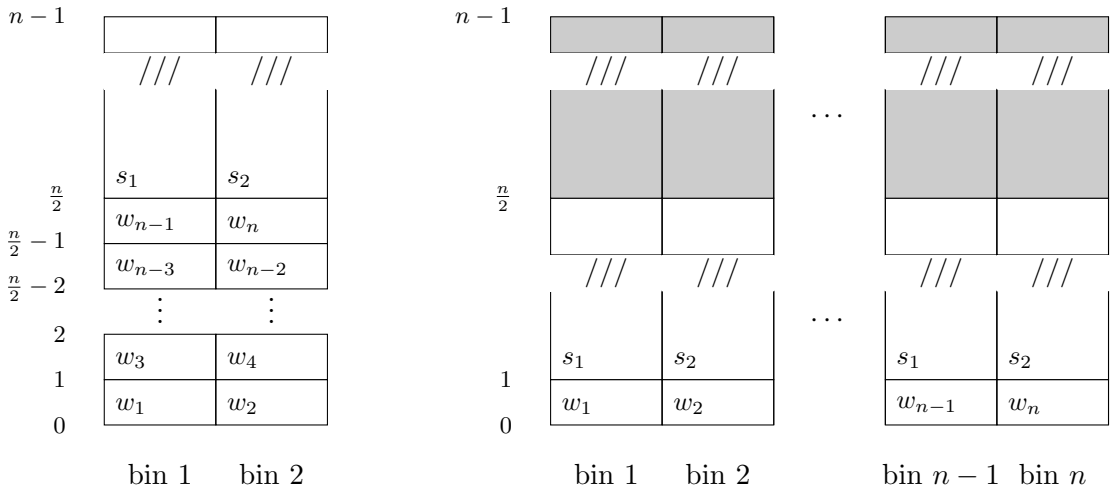
Assume the items are ordered so that their class labels alternate as $1, 2, 1, 2, \ldots$. After packing the first item (class 1), the residual capacity is

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

The next item (class 2) would require $1 + s_2 = \frac{n}{2} > \frac{n}{2} - 1$ 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,

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

□



(a) An optimal BPPS solution

(b) The NF heuristic solution

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

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 1 is given in Figure 1.

We continue with FF and BF, exhibiting a family of BPPS instances on which these heuristics open a number of bins that grows linearly with the number of items, whereas an optimal BPPS solution packs all items into only three bins.

Proposition 2. *For the* BPPS*, the worst-case performance ratios of* FF *and* BF *are unbounded.*

Proof. Let n be divisible by 6 and consider the BPPS instance I_n with $m = 3$ classes, $|\mathcal{I}_1| = |\mathcal{I}_2| = |\mathcal{I}_3| = n/3$, and

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

Each class fits in one bin 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, $\psi(I_n) \leq 3\,r$. 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 be packed in different bins; with only two bins, they would each be full (load d), leaving no capacity for any item of class 3. Hence $\psi(I_n) \geq 3\,r$, and therefore $\psi(I_n) = 3\,r$.

Assume the items are ordered so that their class labels follow the pattern $(1, 2, 3, 3)$, 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 pattern, FF and BF 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. Hence, when processing each repetition of the pattern the algorithm opens two new bins.

After $n/6$ repetitions of the pattern, $n/3$ 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 $\mathrm{FF}(I_n) = \mathrm{BF}(I_n) = \left(\frac{n}{3} + 2\right)r$, and

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

□

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

Note that in Propositions 1 and 2 all items have unit weight. Therefore, sorting them by non-increasing weight does not induce any meaningful order (it only affects tie-breaking). It follows that the family of instances we proposed are worst-case also for the decreasing-order variants NFD, FFD, and BFD. Thus:

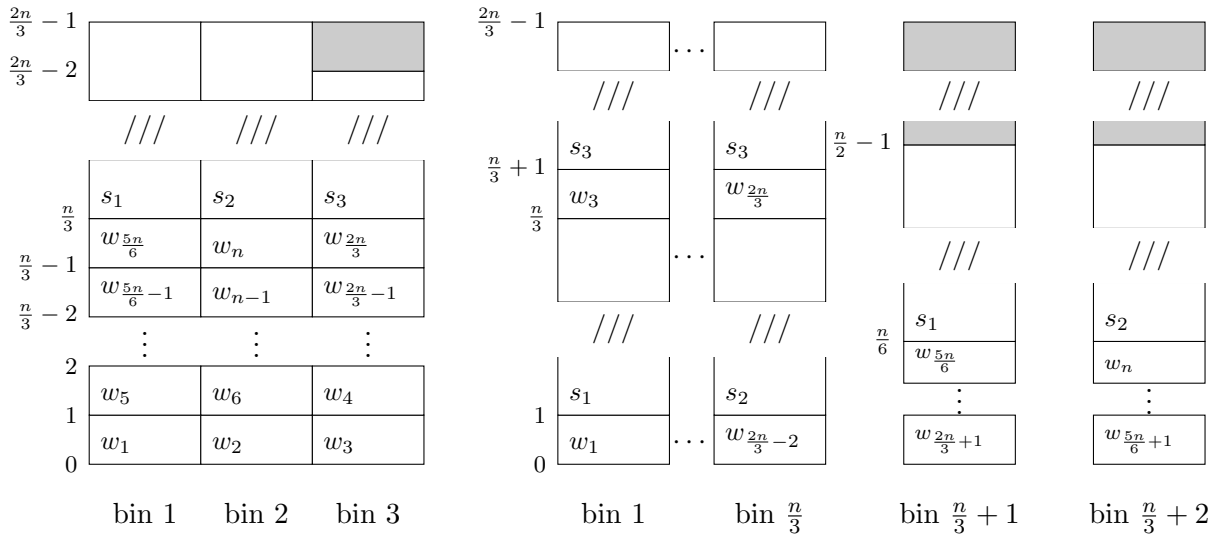
Corollary 1. *For the* BPPS*, the worst-case performance ratios of* NFD*,* FFD*, and* BFD *are unbounded.*

4. Approximation via class-wise optimization and merging

In this section, we show that a constant-factor approximation algorithm for the BPPS can be obtained by applying any constant-factor approximation algorithm for the BPP *class by class*, followed by a *merging phase* of the open bins. We first introduce the necessary notation.

For any subset of items $S \subseteq \mathcal{I}$, let $\mathcal{C}(S) := \{\, c \in \mathcal{C} : S \cap \mathcal{I}_c \neq \emptyset \,\}$ be the *subset of classes* of the items in S, and define its *load* and *cost* as

$$\ell(S) := \sum_{i \in S} w_i + \sum_{c \in \mathcal{C}(S)} s_c, \qquad \kappa(S) := r + \sum_{c \in \mathcal{C}(S)} f_c.$$



(a) An optimal BPPS solution (b) The FF/BF heuristic solution

Figure 2: Worst-case family of instances for FF and BF used in the proof of Proposition 2 (illustrated for the case in which $n/3$ is even). The items are ordered as suggested in the proof. Left: an optimal BPPS solution uses three bins, one per class. Right: FF and BF open a new bin for every item of classes 1 and 2; items of class 3 are packed into the residual space of the first $n/3$ bins, while the remaining bins stay partially empty.

A feasible BPPS solution for an instance I is a family of nonempty subsets $\mathcal{B} \subseteq 2^{\mathcal{I}}$ forming a partition of $\mathcal{I}$ (i.e., $\bigcup_{B\in\mathcal{B}} B = \mathcal{I}$, $B \cap B' = \emptyset$ for all $B, B' \in \mathcal{B}$); feasibility requires $\ell(B) \le d$ for all $B \in \mathcal{B}$. The cost of $\mathcal{B}$ is

$$\overline{\psi}(\mathcal{B}) \;:=\; \sum_{B\in\mathcal{B}} \kappa(B) \;=\; r\,|\mathcal{B}| \;+\; \sum_{B\in\mathcal{B}} \sum_{c\in\mathcal{C}(B)} f_c.$$

For any feasible BPPS solution $\mathcal{B}$, we have $\overline{\psi}(\mathcal{B}) \ge \psi(I)$, hence $\overline{\psi}(\mathcal{B})$ is an upper bound on $\psi(I)$.

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 $d_c := d - s_c$. A feasible BPP solution for I_c is a family of nonempty subsets $\mathcal{B} \subseteq 2^{\mathcal{I}_c}$ forming a partition of $\mathcal{I}_c$ (i.e., $\bigcup_{B\in\mathcal{B}} B = \mathcal{I}_c$, $B \cap B' = \emptyset$ for all $B, B' \in \mathcal{B}$); feasibility requires $\sum_{i\in B} w_i \le d_c$ for all $B \in \mathcal{B}$. Given a BPP algorithm $\mathcal{A}$, we denote by $\mathcal{B}(\mathcal{A}, I_c)$ the solution returned by $\mathcal{A}$ on instance I_c.

We now describe a two-phase heuristic for the BPPS, denoted by TP($\mathcal{A}$), based on a BPP algorithm $\mathcal{A}$. Given an instance I of the BPPS, the heuristic works as follows:

Phase 1. For each $c \in \mathcal{C}$, run $\mathcal{A}$ on I_c and let $\mathcal{B}(\mathcal{A}, I_c)$ be the returned feasible BPP solution. Set

$$\mathcal{B}_1 \;:=\; \bigcup_{c\in\mathcal{C}} \mathcal{B}(\mathcal{A}, I_c).$$

Phase 2. Set $\mathcal{B} := \mathcal{B}_1$. While there exist distinct $B, B' \in \mathcal{B}$ with $\ell(B \cup B') \le d$, replace B and B' by the single set $B \cup B'$. Let $\mathcal{B}_2$ denote the family obtained at termination, and output $\mathcal{B}_2$.

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

We now state and prove the main approximation results of this section: TP($\mathcal{A}$) yields a constant-factor approximation for the BPPS whenever $\mathcal{A}$ does so for the BPP. The main idea of the proof is to decouple the two contributions to the objective (bin-opening cost and setup costs) and to argue for each of them that the merging phase does not increase it.

Theorem 1. *If $\mathcal{A}$ is an α-approximation algorithm for the* BPP *with $\alpha \ge 1$, then the heuristic* TP($\mathcal{A}$) *is a $2\,\alpha$-approximation algorithm for the* BPPS*.*

Proof. Consider a BPPS instance I, and run algorithm TP($\mathcal{A}$), producing $\mathcal{B}_1$ after Phase 1 and $\mathcal{B}_2$ after Phase 2. By construction, both $\mathcal{B}_1$ and $\mathcal{B}_2$ are feasible BPPS solutions. Since we excluded the case $\ell(\mathcal{I}) \le d$ by assumption on the BPPS instances, it follows that $|\mathcal{B}_2| \ge 2$. Let $\mathcal{B}^\star$ be an optimal BPPS solution to I, so that $\overline{\psi}(\mathcal{B}^\star) = \psi(I)$. Moreover, for each $c \in \mathcal{C}$, let $\mathcal{B}^\star(I_c)$ be an optimal BPP solution to I_c.

We now derive an upper bound on $|\mathcal{B}_2|$. At termination of Phase 2, no pair of distinct sets $B, B' \in \mathcal{B}_2$ satisfies $\ell(B \cup B') \le d$, as, otherwise, B and B' would have been merged. This implies that $\ell(B) + \ell(B') > d$, since $\ell(B \cup B') \le \ell(B) + \ell(B')$). We assume the sets in $\mathcal{B}_2$ to be ordered as $B_1, B_2, \ldots, B_{|\mathcal{B}_2|}$ so that $\ell(B_1) \le \ell(B_k)$ for all $k \in \{2, 3, \ldots, |\mathcal{B}_2|\}$. We observe that $\ell(B_k) \le d/2$ holds at most for one set, namely B_1; furthermore, $\ell(B_1) + \ell(B_2) > d$ and $\ell(B_k) > d/2$ for all $k \ge 2$. Thus

$$\sum_{B\in\mathcal{B}_2} \ell(B) = \sum_{k=1}^{|\mathcal{B}_2|} \ell(B_k) = \ell(B_1) + \ell(B_2) + \sum_{k=3}^{|\mathcal{B}_2|} \ell(B_k) > d + \frac{d}{2}\left(|\mathcal{B}_2| - 2\right) = \frac{d}{2}\,|\mathcal{B}_2|,$$

and hence

$$|\mathcal{B}_2| \;\le\; \frac{2}{d} \sum_{B\in\mathcal{B}_2} \ell(B). \tag{1}$$

Merging does not increase total load because $\ell(B \cup B') \le \ell(B) + \ell(B')$ for any pair of sets B, B' in $\mathcal{B}_1$. Therefore

$$\sum_{B\in\mathcal{B}_2} \ell(B) \;\le\; \sum_{B\in\mathcal{B}_1} \ell(B). \tag{2}$$

Since $\mathcal{A}$ is an α-approximation algorithm for the BPP, for each $c \in \mathcal{C}$ we have $|\mathcal{B}(\mathcal{A}, I_c)| \le \alpha\,|\mathcal{B}^\star(I_c)|$. Moreover, each $B \in \mathcal{B}(\mathcal{A}, I_c)$ contains only items of class c, hence $\mathcal{C}(B) = \{c\}$ and $\ell(B) = \sum_{i\in B} w_i + s_c$. It follows that

$$\sum_{B\in\mathcal{B}_1} \ell(B) = \sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}(\mathcal{A}, I_c)| \le \alpha \left(\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}^\star(I_c)| \right). \tag{3}$$

Combining (1), (2), and (3), we have:

$$|\mathcal{B}_2| \leq 2\,\alpha\,\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}^\star(I_c)|\right). \tag{4}$$

The quantity $\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}^\star(I_c)|$ is a lower bound on the total load of any feasible BPPS solution, hence on that of $\mathcal{B}^*$ as well. Hence:

$$\sum_{B\in\mathcal{B}^*}\ell(B) = \sum_{B\in\mathcal{B}^*}\left(\sum_{i\in B} w_i + \sum_{c\in\mathcal{C}(B)} s_c\right) = \sum_{i\in\mathcal{I}} w_i + \sum_{B\in\mathcal{B}^*}\sum_{c\in\mathcal{C}(B)} s_c \geq \sum_{i\in\mathcal{I}} w_i + \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}^\star(I_c)|.$$

Therefore, since $\sum_{B\in\mathcal{B}^\star}\ell(B) \leq d\,|\mathcal{B}^\star|$, we have that $\frac{1}{d}\left(\sum_{i\in\mathcal{I}} w_i \;+\; \sum_{c\in\mathcal{C}} s_c\,|\mathcal{B}^\star(I_c)|\right) \leq |\mathcal{B}^*|$. Thus, from (4) we deduce:

$$|\mathcal{B}_2| \leq 2\,\alpha\,|\mathcal{B}^*|. \tag{5}$$

We now derive an upper bound on the setup-cost contribution. Merging cannot increase it, since $\mathcal{C}(B\cup B') \leq \mathcal{C}(B)\cup\mathcal{C}(B')$, and thus $\sum_{c\in\mathcal{C}(B\cup B')} f_c \leq \sum_{c\in\mathcal{C}(B)} f_c + \sum_{c\in\mathcal{C}(B')} f_c$. Therefore,

$$\sum_{B\in\mathcal{B}_2}\sum_{c\in\mathcal{C}(B)} f_c \leq \sum_{B\in\mathcal{B}_1}\sum_{c\in\mathcal{C}(B)} f_c = \sum_{c\in\mathcal{C}} f_c\,|\mathcal{B}(\mathcal{A},I_c)| \leq \alpha\sum_{c\in\mathcal{C}} f_c\,|\mathcal{B}^\star(I_c)|.$$

As above, $\mathcal{B}^\star$ activates class c in at least $|\mathcal{B}^\star(I_c)|$ sets; hence $\sum_{B\in\mathcal{B}^\star}\sum_{c\in\mathcal{C}(B)} f_c \geq \sum_{c\in\mathcal{C}} f_c\,|\mathcal{B}^\star(I_c)|$, and therefore

$$\sum_{B\in\mathcal{B}_2}\sum_{c\in\mathcal{C}(B)} f_c \leq \alpha\sum_{B\in\mathcal{B}^\star}\sum_{c\in\mathcal{C}(B)} f_c. \tag{6}$$

We can now conclude the proof by combining (5) and (6).

$$\overline{\psi}(\mathcal{B}_2) = r\,|\mathcal{B}_2| + \sum_{B\in\mathcal{B}_2}\sum_{c\in\mathcal{C}(B)} f_c \leq 2\,\alpha\,r\,|\mathcal{B}^\star| + \alpha\sum_{B\in\mathcal{B}^\star}\sum_{c\in\mathcal{C}(B)} f_c \leq 2\,\alpha\,\overline{\psi}(\mathcal{B}^\star) = 2\,\alpha\,\psi(I). \qquad \square$$

By instantiating TP($\mathcal{A}$) with a polynomial-time 3/2-approximation algorithm for the BPP (which is best possible unless $\mathcal{P}=\mathcal{NP}$), such as FFD and BFD [5], we obtain the following approximability result for the BPPS:

Corollary 2. TP(FFD) *and* TP(BFD) *are* 3*-approximation algorithms for the BPPS.*

References

[1] R. Baldacci, F. Ciccarelli, S. Coniglio, V. Dose, and F. Furini. The Bin Packing Problem with Setups: Formulation, Structural Properties and Computational Insights. *Optimization Online*, 2025.

[2] 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.

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

[4] G. Galambos, S. Martello, and D. Vigo. An updated overview of bin packing approximation algorithms. In *Encyclopedia of Algorithms*, pages 1–85. Springer, 2025.

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