Approximating Robust Bin Packing with Budgeted Uncertainty

Aniket Basu Roy[1], Marin Bougeret[1], Noam Goldberg[2], and Michael Poss[1(✉)]

[1] LIRMM, University of Montpellier, CNRS, Montpellier, France
{aniket.basu-roy,marin.bougeret,michael.poss}@lirmm.fr
[2] Department of Management, Bar-Ilan University, 5290002 Ramat Gan, Israel
noam.goldberg@biu.ac.il

Abstract. We consider robust variants of the bin-packing problem where the sizes of the items can take any value in a given uncertainty set $U \subseteq \times_{i=1}^{n}[\overline{a}_i, \overline{a}_i + \hat{a}_i]$, where $\overline{a} \in [0,1]^n$ represents the nominal sizes of the items and $\hat{a} \in [0,1]^n$ their possible deviations. We consider more specifically two uncertainty sets previously studied in the literature. The first set, denoted U^{Γ}, contains scenarios in which at most $\Gamma \in \mathbb{N}$ items deviate, each of them reaching its peak value $\overline{a}_i + \hat{a}_i$, while each other item has its nominal value $\overline{a}_i$. The second set, denoted U^{Ω}, bounds by $\Omega \in [0,1]$ the total amount of deviation in each scenario. We show that a variant of the next-fit algorithm provides a 2-approximation for model U^{Ω}, and a $2(\Gamma+1)$ approximation for model U^{Γ} (which can be improved to 2 approximation for $\Gamma = 1$). This motivates the question of the existence of a constant ratio approximation algorithm for the U^{Γ} model. Our main result is to answer positively to this question by providing a 4.5 approximation for U^{Γ} model based on dynamic programming.

Keywords: Bin-packing · Robust optimization · Approximation algorithm · Next-fit · Dynamic programming

1 Introduction

Bin packing is the problem of assigning a given set of n items, each item of a specified size, to the smallest number of unit capacity bins. The problem has been the subject of study in an extensive body of research initiated by several publications in the 1970s including the work of Johnson et al. [11]. The problem is $\mathcal{NP}$-hard and in fact a straightforward reduction from the partition decision problem implies that it is $\mathcal{NP}$-hard to determine whether a bin-packing instance has a solution using only two bins. This also shows that the problem cannot be approximated within a factor less than 3/2. An approximation factor guarantee of 3/2 has been proven for the first-fit decreasing algorithm by Simchi-Levi [16].

This research has benefited from the support of the ANR project ROBUST [ANR-16-CE40-0018].

Z. Friggstad et al. (Eds.): WADS 2019, LNCS 11646, pp. 71–84, 2019.
https://doi.org/10.1007/978-3-030-24766-9_6

Much of the research has concentrated on the asymptotic setting where n tends to infinity, and in the online setting where the instance is not given in advance but each item is revealed and packed one at a time. A fully polynomial-time approximation scheme for the offline asymptotic problem is due to Karmarkar and Karp [12]. The best asymptotic and absolute online competitive ratios of 1.578 and 5/3, respectively, are due to Balogh et al. in [4] and [3], respectively.

In many applications, the sizes of the items to be packed are not fully known at the time that the packing is carried out. In cargo shipping, for example, the actual weight of a container may deviate from its declared weight or its measurements may be inaccurate. Bin packing has also been used to model the assignment of elective surgeries to operating room in hospitals [8]. Here a bin is a shift of a properly equipped and staffed operating room for performing a certain type of elective surgeries. The room scheduler has to fit in the bins as many cases (patients) as possible. In this setting clearly the length of time of performing each surgery is subject to uncertainty for example in the event of complications. One way to model the uncertainty that falls into the framework of robust optimization is to assume that the sizes are uncertain parameters taking any value in a given set $U \subset \mathbb{R}^n$, where each $a \in U$ represents a possible scenario. This leads to the following problem (where the description of U is sometimes not explicit to avoid exponential length in n)

RBP (Robust bin-packing)
Input: $U \subset \mathbb{R}^n$
Output: A solution is a partition of $[n]$ into k bins $b_1, \ldots, b_k$ such that $\max_{a \in U} \sum_{i \in b_j} a_i \leq 1$ for each $j \in [k]$
Minimize: k

Classically, robust combinatorial optimization has dealt with uncertain objective, meaning that the cost vector c can take any value in set U, unlike RBP where the uncertainty affects the feasibility of the solutions. In that context, it is well-known that arbitrary uncertainty sets U lead to robust counterparts that are hardly approximable. For instance, the robust knapsack is not approximable at all [1], while the shortest path, the spanning tree, the minimum cut, and the assignment problem do not admit constant-ratios approximation algorithms, e.g. [13,14]. Furthermore, describing U by an explicit list of scenarios runs the risk of over-fitting so the optimal solutions may become infeasible for small variations outside U. These two drawbacks are usually tackled by using more specific uncertainty sets, defined by simple budget constraints. One of these widely used uncertainty sets, U^Γ, supposes that the size of each item is either its given nominal size $\bar{a}_i$, or its peak value $\bar{a}_i + \hat{a}_i$. Furthermore, in any scenario, at most $\Gamma \in \mathbb{N}$ of the items may assume their peak value simultaneously. Formally, U^Γ can be defined as $U^\Gamma = \{a | \forall i \in [n], a_i \in \{\bar{a}_i, \bar{a}_i + \hat{a}_i\}$ and $\sum_{i \in [n]} (a_i - \bar{a}_i)/\hat{a}_i \leq \Gamma\}$.[1] Set U^Γ has been widely used in robust combinatorial optimization with a constant

[1] U^Γ is often defined alternatively in the literature, as the polytope $\{a \in \times_{i \in [n]} [\bar{a}_i, \bar{a}_i + \hat{a}_i] \mid \sum_{i \in [n]} (a_i - \bar{a}_i)/\hat{a}_i \leq \Gamma\}$. For the bin-packing problem, one readily verifies using classical arguments that the two definitions lead to the same optimization problem.

number of constraints because the set essentially preserves the complexity and approximability properties of the nominal problem. The result was initially proposed for min-max problems in [6], and was independently extended to uncertain constraints in [2,9], contrasting with the aforementionned uncertain objective. We also consider a second uncertainty set (used in [10,18], among others), characterized again by $\bar{a}$ and $\hat{a}$, as well as the number $\Omega \in [0,1]$ stating how much deviation can be spread among all sizes, formally $U^{\Omega} = \{a \in \times_{i\in[n]}[\bar{a}_i, \bar{a}_i + \hat{a}_i] \mid \sum_{i\in[n]}(a_i - \bar{a}_i) \leq \Omega\}$. From the approximability viewpoint, set U^{Ω} benefits from similar positive results as U^{Γ}, see [15].

The above positive complexity results (e.g. [9,15]) imply, for instance, that there exists a fully-polynomial time approximation scheme (FPTAS) for the robust knapsack problem with uncertain profits and uncertain weights belonging to U^{Ω} and/or U^{Γ}. Interestingly, these positive results do not extend to most scheduling problems (because they involve non-linearities) and to the bin-packing problem (because it involves a non-constant numbers of robust constraints). While in a previous paper [7] (with authors in common) we provided approximability results on robust scheduling, no such results have yet been proposed for the bin-packing problem, the only previous work focusing on numerical algorithms [17]. The purpose of this paper is to fill these gaps, as we present constant-ratio approximation algorithms the bin-packing problem, both for U^{Ω} and U^{Γ}.

Notations, Problems Definitions, and Next-Fit Algorithm. In this paper we consider two special cases of RBP. In the first one, ΓRBP, the input is $\mathcal{I} = (n, \bar{a}, \hat{a}, \Gamma)$ where $n \in \mathbb{N}$, and we assume that $U = U^{\Gamma}$. In the second one, ΩRBP, the input is $\mathcal{I} = (n, \bar{a}, \hat{a}, \Omega)$ where $n \in \mathbb{N}$, $\bar{a} \in [0,1]^n$, $\hat{a} \in [0,1]^n$, and $\Omega \in [0,1]$, and we assume that $U = U^{\Omega}$.

Let us now provide some important notations that will allow us to restate ΓRBP and ΩRBP in a more convenient way. Given $n \in \mathbb{N}$, sets $\{0,1,\ldots,n\}$ and $\{1,\ldots,n\}$ are respectively denoted $[n]_0$ and $[n]$. Set $\{i,\ldots,j\}$ is denoted by $[\![i,j]\!]$. Given a vector $v \in [0,1]^n$ and a subset $X \subseteq [n]$, we define $v(X) = \sum_{i\in X} v_i$. Given two vectors $\bar{a} \in [0,1]^n$, $\hat{a} \in [0,1]^n$ and a subset of items $X \subseteq [n]$, we define $\hat{a}_{\Omega}(X) = \min\{\hat{a}(X), \Omega\}$, $\Gamma(X)$ as the set of Γ items in X with largest $\hat{a}$ values (ties broken by taking smallest indices), or $\Gamma(X) = X$ if $|X| < \Gamma$, and $\hat{a}_{\Gamma}(X) = \hat{a}(\Gamma(X))$. Accordingly, we define the fill of a bin $b \subseteq [n]$ as $f_{\Gamma}(b) = \bar{a}(b) + \hat{a}_{\Gamma}(b)$ for set U^{Γ}, and $f_{\Omega}(b) = \bar{a}(b) + \hat{a}_{\Omega}(b)$ for set U^{Ω}. The fill of a bin for a general uncertainty set U is denoted as $f_U(b) = \max_{a\in U} a(b)$.

Consider the following example. We are given an ordered set of pairs $(\bar{a}_i, \hat{a}_i)$, $X = \{(0.3,0.2),(0.4,0.2),(0.3,0.1),(0.2,0.5)\}$ with $\Gamma = 2$ and $\Omega = 0.3$. Thus, $\Gamma(X) = \{(0.3,0.2),(0.2,0.5)\}$, $\bar{a}(X) = 1.2$, $\hat{a}_{\Gamma}(X) = 0.7$, and $f_{\Gamma}(X) = 1.9$. Similarly, $\hat{a}_{\Omega}(X) = 0.3$ and $f_{\Omega}(X) = 1.0$.

Now, observe that $\max_{a\in U} \sum_{i\in b_j} a_i \leq 1$ (the constraint required in RBP) is equivalent to $f_U(b) \leq 1$, and thus to $f_{\Gamma}(b_j) \leq 1$ for ΓRBP and $f_{\Omega}(b_j) \leq 1$ for ΩRBP. For example in ΓRBP, $f_{\Gamma}(b_j) \leq 1$ simply means that the total nominal ($\bar{a}$) size of the items plus the deviating size ($\hat{a}$) of the Γ largest (in $\hat{a}$ values)

items must not exceed one. Thus, the two optimization problems studied in this paper can be equivalently formulated in the following way.

ΓRBP (Γ-robust bin-packing)
Input: $\mathcal{I} = (n, \overline{a}, \hat{a}, \Gamma)$ where $n \in \mathbb{N}$, $\overline{a} \in [0,1]^n$, $\hat{a} \in [0,1]^n$, and $\Gamma \in \mathbb{N}$.
Output: A solution is a partition of $[n]$ into k bins $b_1, \ldots, b_k$ such that $f_\Gamma(b_j) \leq 1$ for each $j \in [k]$
Minimize: k

ΩRBP (Ω-robust bin-packing)
Input: $\mathcal{I} = (n, \overline{a}, \hat{a}, \Omega)$ where $n \in \mathbb{N}$, $\overline{a} \in [0,1]^n$, $\hat{a} \in [0,1]^n$, and $\Omega \in [0,1]$.
Output: A solution is a partition of $[n]$ into k bins $b_1, \ldots, b_k$ such that $f_\Omega(b_j) \leq 1$ for each $j \in [k]$
Minimize: k

The optimal solution value or cost of either problem is denoted by $\mathsf{OPT}(\mathcal{I}) = k^*$ ($\mathcal{I}$ may be omitted when the instance is clear from the context) and a corresponding optimal solution is denoted by $s^* = \{b_1^*, b_2^*, \ldots, b_{k^*}^*\}$. We introduce in Algorithm 1 a variant of the standard next fit algorithm.

initialization: $j = 1$
1 Pack items (with smaller index first) in b_j until $f_U(b_j) > 1$ or $n \in b_j$. If $n \notin b_j$ then $j \leftarrow j + 1$ and repeat Step 1. Otherwise, $k' \leftarrow j$ proceed to Step 2.
2 Pack the last item of each bin in a new bin: for any j, let $i = \max(b_j)$, $b_j^1 = b_j \setminus \{i\}$, and $b_j^2 = \{i\}$
return : $\bigcup_{j=1}^{k'} \{b_j^1, b_j^2\}$

Algorithm 1. NEXT-FIT($\mathcal{I}$)

Structure of the Paper. In Sects. 2 and 3, we analyze the ratio provided by NEXT-FIT for ΩRBP and ΓRBP, respectively. For ΩRBP, using ordering (1) (non-increasing ordering on $\frac{\hat{a}_i}{\overline{a}_i}$) the ratio is equal to 2. For ΓRBP, using ordering (2) (non-increasing ordering on $\hat{a}_i$), the ratio is bounded by $2(\Gamma + 1)$ (and can be improved to 2 for $\Gamma = 1$). As Theorem 4 shows that neither ordering (1) or (2) leads to a constant ratio using NEXT-FIT, this raises the question the existence of a constant approximation for ΓRBP. In Sect. 4 we first review some basic ideas and explain why they are not sufficient. Then, we introduce the key elements necessary to develop our dynamic programming algorithm (DP) in Sect. 5. The latter gives a ratio of 4.5 for ΓRBP and any $\Gamma \in \mathbb{N}$, which is our main result. The complete proofs of Theorems and Lemmas with a **(⋆)** symbol can be found in the full version of this paper [5].

2 Next-Fit for ΩRBP

Unlike the classical bin-packing problem, executing Next-Fit on arbitrarily ordered items can lead to arbitrarily bad solutions. For example, given ϵ with $0 < \epsilon \leq \frac{1}{2n}$, consider an instance with $\Omega = 1 - \epsilon$, and items $((2\epsilon, 0), (0, 1-\epsilon), \ldots, (2\epsilon, 0), (0, 1-\epsilon))$, where item $i \in [n]$ is denoted by the pair $(\bar{a}_i, \hat{a}_i)$. Using this ordering, Next-Fit will create $n/2$ bins b_j with $f_\Omega(b_j) > 1$ for any $j \in [n]$ (which will be turned into n bins $\{b_j^1, b_j^2\}$), whereas the optimal solution uses 2 bins. This example also illustrates that, unlike in the standard bin-packing, the total size argument no longer apply to the robust counterpart as having $f_\Omega(b_j) > 1$ for any $j \in [n]$ does not imply a large (depending on n) lower bound on the optimal.

Next, we consider an ordering of the items such that

$$\hat{a}_1/\bar{a}_1 \geq \cdots \geq \hat{a}_n/\bar{a}_n. \tag{1}$$

Lemma 1. *Suppose that the items are ordered according to* (1)*. Then* $k' \leq k^*$.

Proof. Consider an optimal solution $b_1^*, \ldots, b_{k^*}^*$ and the subset of optimal bins given by $G^* = \{j \in [k^*] \mid \hat{a}(b_j^*) > \Omega\}$. Let

$$A = \sum_{i \in [n]} (\bar{a}_i + \hat{a}_i) = \sum_{j=1}^{k'} (\bar{a}(b_j) + \hat{a}(b_j)) = \sum_{j=1}^{k^*} \left(\bar{a}(b_j^*) + \hat{a}(b_j^*)\right).$$

Let G denote the first $|G^*|$ bins opened in Step 1 of Next-Fit. If $k' \in G$ then clearly $k' \leq k^*$. Otherwise, it can be observed that for each $l \in G$, $\bar{a}(b_l) > 1 - \Omega$ (as $\bar{a}(b_\ell) + \hat{a}_\Omega(b_\ell) > 1$ and $\hat{a}_\Omega(b_\ell) \leq \Omega$) and $1 - \Omega \geq \max_{j \in G^*} \bar{a}(b_j^*)$ (as $f_\Omega(b_j) \leq 1$). Thus, $\sum_{j \in G} \bar{a}(b_j) > \sum_{j \in G^*} \bar{a}(b_j^*)$ and so by the assumed ordering (1) of the items, following a standard knapsack argument, $\sum_{j \in G} \hat{a}(b_j) > \sum_{j \in G^*} \hat{a}(b_j^*)$. Letting $\bar{G} = [k'] \setminus G$ and $\bar{G}^* = [k^*] \setminus G^*$, it follows that

$$\sum_{j \in \bar{G}} (\bar{a}(b_j) + \hat{a}(b_j)) = A - \sum_{j \in G} (\bar{a}(b_j) + \hat{a}(b_j)) \leq$$
$$A - \sum_{j \in G^*} \left(a(b_j^*) + \hat{a}(b_j^*)\right) = \sum_{j \in \bar{G}^*} \left(\bar{a}(b_j^*) + \hat{a}(b_j^*)\right)$$

(equality may hold throughout if $G^* = \emptyset$). Further, for each $j \in \bar{G} \setminus \{k'\}$, $\bar{a}(b_j) + \hat{a}(b_j) \geq f(b_j) > 1$ and for each $j \in \bar{G}^*$, $\bar{a}(b_j^*) + \hat{a}(b_j^*) \leq 1$. Therefore, $|\bar{G}| \leq \left\lceil \sum_{j \in \bar{G}} (\bar{a}(b_j) + \hat{a}(b_j)) \right\rceil \leq \left\lceil \sum_{j \in \bar{G}^*} \left(\bar{a}(b_j^*) + \hat{a}(b_j^*)\right) \right\rceil \leq |\bar{G}^*|$ and $k' \leq k^*$ as claimed. □

The lemma combined with Step 2 of Next-Fit immediately imply the following theorem.

Theorem 1. *If the items are ordered according to* (1) *then* Next-Fit *is a 2-approximation algorithm for* ΩRBP.

3 Next-Fit for ΓRBP

From now on, we focus on problem ΓRBP. Remark first that using an arbitrary ordering leads to arbitrarily bad solutions, considering $\Gamma = 1$ and the same items $((2\epsilon, 0), (0, 1-\epsilon), \ldots, (2\epsilon, 0), (0, 1-\epsilon))$ as in the previous section. Thus, we consider here an ordering of the items such that

$$\hat{a}_1 \geq \cdots \geq \hat{a}_n. \tag{2}$$

The main result of this Section is the following.

Theorem 2 ($\star$). *If the items are ordered according to (2) then* Next-Fit *is a $2(\Gamma+1)$-approximation algorithm for Γ*RBP.

The proof of Theorem 2 can be found in the full version of this paper [5].

We show here a simplified analysis showing that for $\Gamma = 1$, Next-Fit with ordering (2) is a 2-approximation.

The deviating item of bin j in a fixed optimal solution s^* and in the solution of Next-Fit are denoted by singleton sets $\{i_j^*\} = \Gamma(b_j^*)$ and $\{i_j\} = \Gamma(b_j)$, respectively. We order the bins of s^* such that $i_j^* \geq i_{j+1}^*$. Notice that by definition of Next-Fit and ordering (2) we also have $i_j \geq i_{j+1}$.

Lemma 2. *Suppose that the items are ordered according to (2) and that $\Gamma = 1$. Then $k' \leq k^*$.*

Proof. Suppose by contradiction that $k' > k^*$. Let $b_1, \ldots, b_{k'}$ be the bins opened at Step 1 of Next-Fit and notice that $f_\Gamma(b_j) = \overline{a}(b_j) + \hat{a}_\Gamma(b_j) > 1$ for each $j \in [k'-1]$, while $\overline{a}(b_j^*) + \hat{a}_\Gamma(b_j^*) \leq 1$ for each $j \in [k^*]$. We prove next by induction on $\ell \in [k^*]$ that

$$\sum_{j=1}^{\ell} \overline{a}(b_j) > \sum_{j=1}^{\ell} \overline{a}(b_j^*). \tag{3}$$

For $\ell = 1$, we have $i_1 = i_1^* = 1$ and (3) follows immediately from $\hat{a}_\Gamma(b_1) = \hat{a}_\Gamma(b_1^*)$. Suppose now that induction hypothesis is true for $\ell - 1$. By definition of i_ℓ^* and i_ℓ, we know that $[i_\ell^* - 1] \subseteq \bigcup_{j=1}^{\ell-1} b_j^*$ and $[i_\ell - 1] = \bigcup_{j=1}^{\ell-1} b_j$. Using induction hypothesis, we get that $i_\ell \geq i_\ell^*$, and accordingly $\hat{a}_\Gamma(b_\ell) \leq \hat{a}_\Gamma(b_\ell^*)$. As $l \leq k^* < l$, we have $f_\Gamma(b_l) > 1$, leading to $\overline{a}(b_\ell) > \overline{a}(b_\ell^*)$.

Thus, for $l = k^*$ we get $\sum_{j=1}^{k^*} \overline{a}(b_j) > \sum_{j=1}^{k^*} \overline{a}(b_j^*) = \sum_{i \in [n]} \overline{a}_i$, which is impossible. □

As in the previous section, we obtain the following theorem.

Theorem 3. *If the items are ordered according to (2) and $\Gamma = 1$ then* Next-Fit *is a 2-approximation algorithm for Γ*RBP.

To complete the analysis, we establish the following lower bound on the ratio of Next-Fit.

Theorem 4 (⋆). *If the items are ordered according to (2) or (1), then the approximation ratio of* Next-Fit *for Γ*RBP *is at least $\frac{2\Gamma}{3}$.*

Proof. Let us define an instance where the ordering (2) can lead to Step 1 of Next-Fit using $k' = \Gamma$ bins while $\mathsf{OPT} = 3$. Every row of the $\Gamma \times \Gamma$ matrix below corresponds to the set of items in a bin (after the Step 1) of Next-Fit algorithm

$$\begin{array}{cccc} (\epsilon, 1/\Gamma - \delta_1) & (0, 1/\Gamma - \delta_1) & \dots & (0, 1/\Gamma - \delta_1) \\ \vdots & \vdots & \ddots & \vdots \\ (\epsilon, 1/\Gamma - \delta_\Gamma) & (0, 1/\Gamma - \delta_\Gamma) & \dots & (0, 1/\Gamma - \delta_\Gamma) \end{array} \tag{4}$$

where $\epsilon \leq 1/\Gamma$ and $\delta_1 \leq \dots \leq \delta_\Gamma < \epsilon/\Gamma$. On the one hand, $\epsilon + \Gamma \cdot (1/\Gamma - \delta_l) > 1$ for each $l \in [\Gamma]$, so step 1 of Next-Fit outputs Γ bins. On the other hand, an optimal solution can pack all the items above except the ones in the first column into a single bin because $\Gamma \cdot 1/\Gamma - \delta_1 \leq 1$. Further, the total weight of the first $\Gamma/2$ items of the first column sums up to $\Gamma/2 \cdot (1/\Gamma + \epsilon) - \sum_{l=1}^{\Gamma/2} \delta_l \leq 1 - \sum_{l=1}^{\Gamma/2} \delta_l \leq 1$, and similarly for the last $\Gamma/2$ items, so an optimal solution may pack the first column using two bins. Finally, instance (4) shows that Next-Fit produces a solution $2\Gamma/3$ times worse than the optimal one.

This instance can be adapted to establish a lower bound for the approximation ratio of Next-Fit when items are ordered according to (1); see [5]. □

4 First Ideas to Get a Constant Ratio for ΓRBP

We maintain the assumption that the items are ordered according to (2).

4.1 Attempts to Get a Constant Ratio

We discuss below some natural arguments to get constant ratios.

Attempt 1: Using a Classical Size Argument. Next-Fit without a particular ordering applied to instance of Sect. 3 leads to a solution with $k' = n/2$ bins (at the end of Step 1) where $f_\Gamma(b_j) > 1$ for each bin, while $\mathsf{OPT} = 2$. This example shows that even if all bins are "full" (relatively to f_Γ), it does not provide a lower bound on the optimal number of bins. Moreover, as shown in Theorem 4, none of the two orders considered in the previous section leads to a constant ratio using Next-Fit.

Attempt 2: Using the Duality with Makespan Minimization. Given input $\mathcal{I}$, we could guess $k^* = \mathsf{OPT}(\mathcal{I})$, and then consider the input $(\mathcal{I}, k^*)$ as an input of robust makespan minimzation (which was studied in [7]). Using any ρ-approximation for the later problem (for example $\rho = 3$ in [7]), we could get in polynomial time a solution with k^* bins an such that $f_\Gamma(b_j) \leq \rho$. The last step would be to convert this solution into a solution of ΓRBP by unpacking each

bin (with $f_\Gamma(b_j) \leq 3$) into several bins b'_j with $f_\Gamma(b'_j) \leq 1$. However, even if ρ were arbitrarily close to 1, it is not possible to bound (for a fixed j) the number of bins b'_j by a constant as showed in the instance containing n items $(\frac{\epsilon}{n}, 1 - \frac{\epsilon}{n})$ and $\Gamma = 1$. While all items fit into a single bin with capacity lower than $1 + \epsilon$, they require n bins of capacity 1 to be packed.

Attempt 3: Guessing the Profile of an Optimal Solution. Let $\mathcal{I}$ be a input of ΓRBP. Given a solution $s = \{b_j, j \in [k]\}$ for this input, we define $P(s) = \{\Gamma(b_j), j \in [k]\}$ as the profile of s and $\tilde{P}_j(s) = \{i \mid i \in \Gamma(b_\ell)$ for some $\ell \in [j]\}$ as all deviating items in the first j bins. Let $s^* = \{b^*_j, j \in [k^*]\}$ be an optimal solution. To get some insight on the problem, let us assume that we know $P(s^*)$ (even if this cannot be guessed in polynomial time). We show how we can use $P(s^*)$ to get a 2-approximation algorithm. Without loss of generality, we can always assume that $|\Gamma(b^*_j)| = \Gamma$ for any j, as otherwise we can add $\Gamma - |\Gamma(b^*_j)|$ dummy items of size $(0, 0)$ to b^*_j. Remember that the items are sorted in non-increasing order of their deviating values ($\hat{a}_i \geq \hat{a}_{i+1}$). For any $j \in [k^*]$, let $i^*_j = \max(\Gamma(b^*_j))$ be the smallest (in term of $\hat{a}$ value) deviating item of bin j (when $\Gamma = 1$, $\{i^*_j\} = \Gamma(b^*_j)$ as in the previous section). Without loss of generality, let us assume that bins are sorted such that $i^*_j \geq i^*_{j+1}$. Now, given $P(s^*)$, in the first phase we construct a solution s by packing items of $P(s^*)$ as they were packed in s^*, meaning that we define $b_j = \Gamma(b^*_j)$ for $j \in [k^*]$. Let $X = [n] \setminus \bigcup_{j \in [k^*]} \Gamma(b^*_j)$ be the set of remaining items. We now pack X in the following second phase, starting with $j = 1$. Notice that in the description of the algorithm below, we consider that for $j \in [k^*]$, b_j already contains $\Gamma(b^*_j)$, whereas for any $j > k^*$, b_j is initially empty.

Step 1 pack items of X (by decreasing $\hat{a}$ values) in b_j until $f_\Gamma(b_j) > 1$ or $X = \emptyset$
Step 2 if $X \neq \emptyset$, $j = j + 1$, and go to step 1.

Let j be the bin such that X is empty after filling b_j. Let k' be the number of bins used by this algorithm. Notice that if $j \leq k^*$ then $k' = k^*$ (because of the pre-packing of item of $P(s^*)$), and otherwise $k' = j$.

Lemma 3. *$k' \leq k^*$, implying a 2-approximation as we can convert the solution of* Next-Fit *into a feasible solution of $2k'$ bins by repacking the last added item in each bin in a separate bin.*

Proof. Assume by contradiction that $k' > k^*$. Informally, as an item $i \in [n] \setminus \tilde{P}_{k^*}(s^*)$ does not deviate in s^*, we need to ensure that this is also the case in s. Let us prove by induction on j that the items packed greedily in Step 1 satisfy

$$\hat{a}_i \leq \hat{a}_{i^*_j}, \forall i \in b_j \setminus \tilde{P}_{k^*}(s^*), j \in [k^*]. \tag{5}$$

Let $j = 1$, and suppose there is $i \in b_1 \setminus \tilde{P}_{k^*}(s^*)$ such that $\hat{a}_i > \hat{a}_{i^*_1}$. Then, because $\hat{a}_{i^*_1} \geq \hat{a}_{i^*_j}$ for $j > 1$, $\hat{a}_i > \hat{a}_{i^*_j}$ for each j so $i \in \tilde{P}_{k^*}(s^*)$, a contradiction.

Now, consider bin b_{j+1}. By induction, we have that $\sum_{\ell\in[j]} \overline{a}(b_\ell) + \hat{a}(\tilde{P}_j(s)) > j \geq \sum_{\ell\in[j]} \overline{a}(b^*_\ell) + \hat{a}(\tilde{P}_j(s^*))$, so $\tilde{P}_j(s) = \tilde{P}_j(s^*)$ implies

$$\sum_{\ell\in[j]} \overline{a}(b_\ell) > \sum_{\ell\in[j]} \overline{a}(b^*_\ell). \tag{6}$$

Let X_j be the set of items of X left after packing bin b_j by the above procedure and X^*_j be the the set of items of X left after the optimal solution packs bin b^*_j. Inequality (6) and the ordering used in Step 1 imply that $\lambda \geq \lambda^*$, where $\lambda = \min(X_j)$ and $\lambda^* = \min(X^*_j)$. Therefore, if there exists $i \in b_{j+1} \setminus \tilde{P}_{k^*}(s^*)$ such that $\hat{a}_i > \hat{a}_{i^*_{j+1}}$, then $\hat{a}_{\lambda^*} \geq \hat{a}_\lambda \geq \hat{a}_i > \hat{a}_{i^*_{j+1}}$, and thus $\hat{a}_{\lambda^*} > \hat{a}_{i^*_\ell}$ for any $l \in [\![j+1, k^*]\!]$, which is a contradiction as item λ^* is in X and thus does not deviate in the considered optimal.

Now that Property (5) is proved, let us get our contradiction from $k' > k^*$. Indeed, if $k' > k^*$ then $\sum_{i\in[n]} \overline{a}_i > k^* - \hat{a}(\tilde{P}_{k^*}(s^*)) \geq \sum_{j\in[k^*]} \overline{a}(b^*_j)$ where the first inequality follows from $f_\Gamma(b_j) > 1$ for $j \in [k^*]$ and Property (5), and the second one follows from $\sum_{j\in[k^*]} \overline{a}(b^*_j) + \hat{a}(\tilde{P}_{k^*}(s^*)) \leq k^*$. This implies a contradiction. □

Even if the above procedure relies on a guessing step which is not polynomial, its core idea has similarities with both the analysis of Next-Fit in the proof of Theorem 2 (see [5]) and with the DP algorithm detailed later in this paper, where we only guess the deviating item with the smallest deviation of each bin (one at a time), and we pack $\Gamma - 1$ items "better" than the one packed in $P(s^*)$, at the expense of a few extra bins.

4.2 Restricting Our Attention to Small Items

We define ΓRBP with small values as the ΓRBP problem restricted to inputs where for any $i \in [n]$, $\hat{a}_i \leq \frac{1}{\Gamma}$ and $\hat{a}_i \leq \frac{1}{\Gamma}$. Below we give a justification for restricting our attention to ΓRBP with small values.

Lemma 4. *Any polynomial ρ-approximation for ΓRBP with small values implies a polynomial $(\rho + \rho_{bp})$-approximation for ΓRBP, where ρ_{bp} is the best known ratio of a polynomial time approximation for classical bin-packing.*[2]

Proof. Given an instance $\mathcal{I}$ of ΓRBP, we define the small items $\mathcal{S} = \{i \in [n] : \overline{a}_i \leq 1/\Gamma \text{ and } \hat{a}_i \leq 1/\Gamma\}$ and the large item as $\mathcal{B} = [n] \setminus \mathcal{S}$. We use our ρ-approximation algorithm to pack $\mathcal{S}$ into $k_\mathcal{S}$ bins, implying $k_\mathcal{S} \leq \rho\mathsf{OPT}(\mathcal{S}) \leq \rho\mathsf{OPT}(\mathcal{I})$. Then, we observe that in any packing of $\mathcal{B}$, each bin contains no more than Γ items, so that all items deviate in these bins. Hence, ΓRBP for instance $(\mathcal{B}, \Gamma)$ is equivalent to the classical bin-packing problem for items $\mathcal{B}'$

[2] In general, if we have a polynomial time additive approximation algorithm using $OPT + f(OPT)$ bins and polynomial time ρ-approximation algorithm for ΓRBP with small values then our algorithm uses $OPT(\rho+1) + f(OPT)$ bins for ΓRBP in polynomial time.

where the weight of each item $i \in \mathcal{B}'$ is given by $\overline{a}_i + \hat{a}_i$. This implies that $\mathsf{OPT}_{bp}(\mathcal{B}') = \mathsf{OPT}(\mathcal{B})$ (where OPT_{bp} denotes the optimal value in classical bin-packing), and that any solution for $\mathcal{B}'$ is a solution of $\mathcal{B}$. Thus, we use a ρ_{bp}-approximation algorithm for classical bin-packing to pack $\mathcal{B}'$ in $k_{\mathcal{B}}$ bins, and use the same packing for items in $\mathcal{B}$. Note that $k_{\mathcal{B}} \leq \rho_{bp}\mathsf{OPT}_{bp}(\mathcal{B}) = \rho_{bp}\mathsf{OPT}(\mathcal{B}) \leq \rho_{bp}\mathsf{OPT}(\mathcal{I})$. We obtain a packing of $\mathcal{I}$ with cost $k_{\mathcal{S}} + k_{\mathcal{B}} \leq (\rho + \rho_{bp})\mathsf{OPT}(\mathcal{I})$. □

Observation 1. *Given an instance $\mathcal{I}$ to the ΓRBP with small values, any subset $X \subseteq [n]$ can be packed in $\lceil \frac{|X|}{\Gamma/2} \rceil$ bins.*

Notice that instances with small items are not easier to approximate by Next-Fit because instance (4) from Sect. 3 uses small items.

4.3 Guessing of the Full Profile and Considering Only Small Items

Let us now explain why mixing the two previous ideas is promising. As in attempt 3 where we know the full profile, we want to construct for any j bins $\{b_1, \ldots, b_j\}$ such that their total $\overline{a}$ is larger than the total value of $\overline{a}$ packed by the first j bins of s^* (the considered optimal solution), as in inequality (6). Instead of guessing the full profile $P(s^*)$, we want to design a DP algorithm (that guesses i_{j^*} one at the time) with the following intuitive outline. Start with $j = 1$.

- guess item i_j^*, the smallest (in $\hat{a}$ value) deviating item of b_j^*, and pack it in b_j
- then, as the $\Gamma - 1$ other deviating items in b_j^* are unknown and we want to pack more of the nominal size $\overline{a}$, packs separately $\Gamma - 1$ items with larger $\overline{a}$ values (among items with $\hat{a}$ values greater than $\hat{a}_{i_j^*}$). Consider that these $\Gamma - 1$ items are put in the "trash" (at the very end we will pack all items of the trash in a few additional bins)
- keep filling bin b_j greedily (by non-increasing $\hat{a}$ values) until exceeding 1
- make a recursive call with $j + 1$

If s^* uses k^* bins, we wish to output a solution s with k^* bins exceeding one, and $(\Gamma - 1)k^*$ items in the trash. This almost feasible solution can be converted into a regular one with $3k^*$ bins by removing one item from each bin and adding them to the trash, and packing the Γk^* items of the trash into $2k^*$ bins, which is possible according to Observation 1. This sketches the core ideas of the DP. However, the actual DP presented below needs to be more involved for the following reasons. Consider $j = 1$ for convenience and let $B = [\![1, i_1^* - 1]\!]$.

First, notice that items of B could be packed (as deviating items) in a bin other than b_1^* in s^*, and we may have $|B| > \Gamma - 1$. Thus, instead of trashing only $\Gamma - 1$ items of B, we have to trash all of them, and count the number of trashed items to ensure that at the end at most $(\Gamma - 1)k^*$ items are trashed. To summarize, the trash will represent the union of the $(\Gamma - 1)$ larger (in $\hat{a}$ values) deviating items of each bin. Moreover, we want to maintain that the accumulated nominal ($\overline{a}$) size of trashed items in s is larger than the accumulated nominal size of deviating items in s^*.

Second, notice that in s^*, items of $[\![i_1^* + 1, i_2^* - 1]\!]$ are either in b_1^* as non-deviating items or in a b_j^*, $j \geq 2$ as deviating items (meaning that they are trashed items in s). Thus, if we incorrectly pack some of these items in b_1 instead of trashing them, these items will not be available when considering b_2, and we may not be able to ensure then that trashed items in s have a larger $\overline{a}$ value than the deviating items in s^*.

In the next section we describe the full version of the DP. To that end, we first need to introduce formally the notion of trash.

5 Approximating ΓRBP with Small Values

Bin-Packing with Trash. For any $X \subseteq [n]$, we define $\tilde{a}_\Gamma(X) = \Gamma\hat{a}_1(X)$ ($\tilde{a}_\Gamma(X)$ is Γ times the largest deviating value of an item in X) and $\tilde{f}(X) = \overline{a}(X)+\tilde{a}_\Gamma(X)$. We introduce next a decision problem ΓRBP-T related to ΓRBP.

> ΓRBP-T (Robust bin-packing with trash)
> *Input:* $(\mathcal{I}, k, t)$ where $\mathcal{I}$ is an input of ΓRBP (where each item $(\overline{a}_i, \hat{a}_i)$ satisfies $\hat{a}_i \leq 1/\Gamma$ and $\overline{a}_i \leq 1/\Gamma$), and k, t are two integers.
> *Output:* Decide if a solution exists, where a solution is a partition of the set of items into $k+1$ sets $b_1, \ldots, b_k$ and T (called the trash) such that:
> – $\tilde{f}(b_j) \leq 1$ for each $j = 1, \ldots, k$
> – $|T| \leq t$

Notice that although each item is small in ΓRBP-T, it is possible to have an item i such that $\tilde{f}(\{i\}) > 1$, implying that i must be put in the trash. We show below how deciding ΓRBP-T is enough to approximate ΓRBP.

Lemma 5. *For any input $\mathcal{I}$ of ΓRBP and $k^* = \mathsf{OPT}(\mathcal{I})$, $(\mathcal{I}, k^*, (\Gamma-1)k^*)$ is a yes input of ΓRBP-T.*

Proof. Given an optimal solution of size k^* of ΓRBP problem we create a solution to ΓRBP-T problem as follows. Let b_j^* be a bin of the considered optimum. Let N_j be the non-deviating items of b_j^*, i.e., $b_j^* = N_j \cup \Gamma(b_j^*)$. Let $X = \max(\Gamma(b_j^*))$ (the smallest deviating item of b_j^*) if $|\Gamma(b_j^*)| = \Gamma$ and $X = \emptyset$ otherwise. We define $b'_j = N_j \cup X$, and add items of $Y = b_j^* \setminus b'_j$ into the trash. Notice Y is either the set of $\Gamma - 1$ largest deviating object of b_j^*, or is equal to $\Gamma(b_j^*)$ when $|\Gamma(b_j^*)| < \Gamma$. This is a feasible solution for ΓRBP-T problem as $\tilde{f}(b'_j) = \overline{a}(b'_j) + \tilde{a}_\Gamma(b'_j)$, where $\overline{a}(b'_j) \leq \overline{a}(b_j^*)$ and $\tilde{a}_\Gamma(b'_j) = \tilde{a}_\Gamma(X) \leq \hat{a}_\Gamma(b_j^*)$, and as there are at most $(\Gamma-1)k^*$ items in the trash. □

Lemma 6. *For any input $\mathcal{I}$ of ΓRBP and integer k, given a solution of $(\mathcal{I}, k, \Gamma k)$ of ΓRBP-T, we can compute in polynomial time a solution of $3k$ bins for $\mathcal{I}$.*

Proof. Given a solution $b_1, \ldots, b_k$, T for $(\mathcal{I}, k, \Gamma k)$ of ΓRBP-T the bins remain feasible in ΓRBP as $f_\Gamma(b_j) = \overline{a}(b_j) + \hat{a}_\Gamma(b_j) \leq \overline{a}(b_j) + \tilde{a}(b_j) = \tilde{f}(b_j)$. Then, Observation 1 implies that the trash T can be packed into $\lceil k\Gamma/(\Gamma/2) \rceil \leq 2k$ additional bins. □

A DP Algorithm for ΓRBP-T. The objective of this section is to define a DP algorithm that will be used to decide the ΓRBP-T problem. To this aim, we define G-ΓRBP-T (generalized robust bin-packing with trash), an optimization problem that the DP algorithm will solve in a relaxed way. To define G-ΓRBP-T, we consider a fixed instance $\mathcal{I}$ of ΓRBP with items ordered according to (2) and an integer k.

G-ΓRBP-T (generalized robust bin-packing with trash)
Input: $\mathcal{I} = (q, t, \ell)$, where $q \in [n]_0$, $t \in [(\Gamma - 1)k]_0$, and $\ell \in [k + 1]$.
Output: A feasible solution s is a partition of $[\![q, n]\!]$ into $k - \ell + 3$ sets (b_j for $j \in [\![\ell, k]\!]$, b_0 and T), such that
- for any $j \in [\![\ell, k]\!]$, $\tilde{f}(b_j) \leq 1$ (the $k - \ell + 1$ regular bins must respect the constraint of ΓRBP-T)
- $|T| \leq t$ (we only allow t items in the trash)
- $\min(b_\ell) = q$ (meaning that the deviating item of b_ℓ is q as items are sorted in non-increasing order of $\hat{a}$ values)

Minimize: $c(s) = \overline{a}(b_0)$ (in bin b_0 we only count $\overline{a}$ values)

The objective of G-ΓRBP-T is to pack a part (defined by $[\![q, k]\!]$) of an ΓRBP-T instance given a fixed budget of resources (the number of bins and the size of the trash) while minimizing the total nominal size of items in the dummy bin b_0. The last constraint (the deviating item of b_ℓ is q) may appear artificial at first sight, but comes from the fact that the DP will guess at each new bin the largest items that should be packed in it, and therefore this constraint ensures that every optimal solution must pack q in b_ℓ as well.

Definition 1 (almost feasible solution). *We say that a bin b* ***exceeds by at most one item*** *iff $\tilde{f}(b) > 1$ and $\tilde{f}(b \setminus \{i\}) \leq 1$ where $i = \max(b)$. Given an input (q, t, ℓ) of* G-ΓRBP-T*, we say that a solution is* ***almost feasible*** *iff all the above constraints of* G-ΓRBP-T *are respected, except that for any $j \in [\![\ell, k]\!]$, we allow that b_j exceeds by at most one item instead of $\tilde{f}(b_j) \leq 1$.*

The relation between G-ΓRBP-T and ΓRBP-T is characterized in the two following lemmas whose proofs can be found in [5].

Lemma 7 ($\star$). *For any $\mathcal{I}$ input of ΓRBP and k such that $(\mathcal{I}, k, (\Gamma - 1)k)$ is a yes input of ΓRBP-T, there exists q and t such that $\mathsf{OPT}(q, t, 1) = 0$.*

Lemma 8 ($\star$). *Let us fix $\mathcal{I}$ an input of ΓRBP and k an integer. For any $q \in [(\Gamma - 1)k], t = (\Gamma - 1)k - (q - 1)$, given an almost feasible solution of $\mathcal{I}' = (q, t, 1)$ of cost 0 for* G-ΓRBP-T*, we can compute in polynomial time a solution of $(\mathcal{I}, k, \Gamma k)$ of ΓRBP-T.*

Thus, Lemmas 6 and 8 show that providing an almost feasible solution for $(q, t, 1)$ of cost 0 for G-ΓRBP-T implies a solution of size $3k$ for ΓRBP.

Let us now define a DP algorithm $DP(\mathcal{I})$ ($\mathcal{I}$ is an input of G-ΓRBP-T) that provides an **almost feasible** solution s with $c(s) \leq \mathsf{OPT}(\mathcal{I})$ (where $\mathsf{OPT}(\mathcal{I})$ is by definition the optimal cost of a **feasible** solution). We provide below a gentle description of the DP. Given an instance $\mathcal{I} = (q, t, \ell)$, the DP starts by guessing (q^*, t^*), where

- $q^* = \min(b^*_{l'})$ for a bin $b^*_{l'}$ with $l' \in [\![l+1, k^*]\!]$ of an optimal solution s^*
- t^* is the number of items trashed from X^* in s^*, where $X^* = [\![q, q^*-1]\!]$
- Notice that in s^* items of X^* must by placed in b^*_l, b^*_0 or T^*. We mimic the optimal in the current call of the DP by packing X^* in b_l, b_0 and T.
- To that end, the DP:
  - packs q in b_l (as required by the corresponding constraint of G-ΓRBP-T),
  - packs the t^* largest (in terms of $\overline{a}$) remaining items of X^* to the trash
  - packs the remaining items of X^* into b_ℓ until $\tilde{f}(b_\ell) > 1$ or $X^* = \emptyset$
  - packs the remaining items of X^* into b_0 until $X^* = \emptyset$

We discuss next where the other items (of $[\![q^*, n]\!]$) are packed. Notice that in s^*, bin b^*_l may contain items of $[\![q^*, n]\!]$, and thus the DP may also have to pack items of $[\![q^*, n]\!]$ into b_l. The key is that the decision of which items of $[\![q^*, n]\!]$ to pack into b_l is not taken at this step of the algorithm but only later (to avoid packing in b_l items of large $\overline{a}$ value that are in the trash in s^*). To allow this decision to be taken later, let Δ_b be the size of the empty space in b_l after packing X^* as described above, and let $b_0^{X^*} = b_0 \cap X^*$. After the previous steps, the DP makes a recursive call to get a solution $\tilde{s}$ that packs $[\![q^*, n]\!]$ into regular bins, the trash, and a dummy bin $\tilde{b}_0$. So far solution $\tilde{s}$ has not used any of the empty space Δ_b. However, we can unpack items from $\tilde{b}_0$ to b_ℓ while ensuring that these items do not deviate in b_ℓ (as all these items have index greater than q).

The formal description of $DP(q, t, \ell)$ and its correctness, stated formally in the following two results, are provided in [5].

Lemma 9 ($\star$). *For any $\mathcal{I}$ input of* G-ΓRBP-T*, $DP(\mathcal{I})$ provides an almost feasible solution of cost at most $\mathsf{OPT}(\mathcal{I})$.*

Lemma 10 ($\star$). *There is a* 3*-approximation for* ΓRBP *with small values running in $\mathcal{O}(n^6 log(n))$.*

By Lemma 4, the following theorem is now immediate using a $\frac{3}{2}$-approximation for classical bin-packing (see for example in [16]) as a black box.

Theorem 5. *There is a* 4.5*-approximation for* ΓRBP *running in $\mathcal{O}(n^6 log(n))$.*

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. **197**(2), 427–438 (2009)
2. Álvarez-Miranda, E., Ljubic, I., Toth, P.: A note on the Bertsimas & Sim algorithm for robust combinatorial optimization problems. 4OR **11**(4), 349–360 (2013). https://doi.org/10.1007/s10288-013-0231-6
3. Balogh, J., Békési, J., Dósa, G., Sgall, J., van Stee, R.: The optimal absolute ratio for online bin packing. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (2015)

4. Balogh, J., Békési, J., Dósa, G., Epstein, L., Levin, A.: A new and improved algorithm for online bin packing. In: Azar, Y., Bast, H., Herman, G. (eds.) ESA. LIPIcs, vol. 112, pp. 5:1–5:14. Dagstuhl, Germany (2018)
5. Basu Roy, A., Bougeret, M., Goldberg, N., Poss, M.: Approximating the robust bin-packing with budget uncertainty (2019). https://hal.archives-ouvertes.fr/hal-02119351
6. Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Math. Program. **98**(1–3), 49–71 (2003)
7. Bougeret, M., Pessoa, A.A., Poss, M.: Robust scheduling with budgeted uncertainty. Discrete Appl. Math. **261**(31), 93–107 (2019)
8. Dexter, F., Macario, A., Traub, R.D.: Which algorithm for scheduling add-on elective cases maximizes operating room utilization? Use of bin packing algorithms and fuzzy constraints in operating room management. Anesthesiology **91**, 1491–1500 (1999)
9. Goetzmann, K.-S., Stiller, S., Telha, C.: Optimization over integers with robustness in cost and few constraints. In: Solis-Oba, R., Persiano, G. (eds.) WAOA 2011. LNCS, vol. 7164, pp. 89–101. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29116-6_8
10. Gounaris, C.E., Wiesemann, W., Floudas, C.A.: The robust capacitated vehicle routing problem under demand uncertainty. Oper. Res. **61**(3), 677–693 (2013)
11. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. **3**(4), 299–325 (1974)
12. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), pp. 312–320, November 1982
13. Kasperski, A., Zieliński, P.: On the approximability of minmax (regret) network optimization problems. Inform. Process. Lett. **109**(5), 262–266 (2009)
14. Kasperski, A., Zielinski, P.: On the approximability of robust spanning tree problems. Theor. Comput. Sci. **412**(4–5), 365–374 (2011). https://doi.org/10.1016/j.tcs.2010.10.006
15. Poss, M.: Robust combinatorial optimization with knapsack uncertainty. Discrete Optim. **27**, 88–102 (2018). https://doi.org/10.1016/j.disopt.2017.09.004
16. Simchi-Levi, D.: New worst-case results for the bin-packing problem. Naval Res. Logist. **41**, 579–585 (1994)
17. Song, G., Kowalczyk, D., Leus, R.: The robust machine availability problem–bin packing under uncertainty. IISE Trans. **50**(11), 997–1012 (2018). https://doi.org/10.1080/24725854.2018.1468122
18. Tadayon, B., Smith, J.C.: Algorithms and complexity analysis for robust single-machine scheduling problems. J. Sched. **18**(6), 575–592 (2015)