Als Manuskript gedruckt

Technische Universität Dresden
Herausgeber: Der Rektor

Mathematical Models and Approximate Solution Approaches for the Stochastic Bin Packing Problem

John Martinovic, Maximilian Selch

Preprint MATH-NM-02-2020

September 2020

Mathematical Models and Approximate Solution Approaches for the Stochastic Bin Packing Problem

J. Martinovic[a,*], M. Selch[a]

[a] *Institute of Numerical Mathematics, Technische Universität Dresden, Germany*

Abstract

We consider the (single-stage) stochastic bin packing problem (SBPP) which is based on a given list of items the sizes of which are represented by stochastically independent random variables. The SBPP requires to determine the minimum number of unit capacity bins needed to pack all the items, such that for each bin the probability of exceeding the capacity is bounded by a given constant. Such calculations are particularly relevant in the field of server consolidation, where (not precisely known) jobs have to be assigned to as few processing units as possible to keep the energy consumption and operational costs low. Although not being limited to this assumption, our theoretical investigations are only exemplified for normally distributed item sizes, especially since real-world characteristics can often be reasonably approximated by that type of distribution. From a mathematical point of view, such item characteristics are particularly challenging as the corresponding SBPP turns out to be a nonlinear integer program. For this scenario, we first derive several (improved) lower and upper bounds and study their worst-case performance metrics. Moreover, we describe various linearization techniques to overcome the drawback of nonlinearity. Finally, all approaches are tested based on extensive computational experiments, involving both randomly generated and real-world instances.

Keywords: Cutting and Packing, Stochastic Bin Packing Problem, Linearization, Lower and Upper Bounds, Server Consolidation, Normal Distribution

1. Introduction

The *bin packing problem (BPP)* (or, alternatively, the *cutting stock problem (CSP)*), is one of the most important representatives in combinatorial optimization and has been extensively examined in a plethora of works over the years, see [28] and references therein. Given a list of $n \in \mathbb{N}$ items, each being characterized by some size $c_i \in (0, 1)$, $i \in I := \{1, \ldots, n\}$, the BPP requires to find the minimum number of unit capacity bins (i.e., $C = 1$ holds) that is able to accommodate all items without violating the capacity constraint of a single bin. One of the earliest scientific mentions dates back to the year 1939 when Kantorovich formulated a first *integer linear program (ILP)* for the BPP, see [47], having serious drawbacks in terms of symmetry and the LP value, at least from today's perspective. During the following decades, three major groups of exact formulations possessing more favorable theoretical properties have been presented: a pattern-based approach [36], a one-cut formulation [31], and a graph-theoretic model [76]. For the sake of completeness, we would like to point out that any of these modeling types has seen numerous improvements since its publication, see [40, 77] for contributions to accelerated column generation or [9, 26, 60] for reduction procedures related to pseudo-polynomial formulations. For further information regarding the classical BPP, we refer the reader to some (by far not exhaustive) surveys [28,

*Corresponding author

Email addresses: john.martinovic@tu-dresden.de (J. Martinovic), maximilian.selch@tu-dresden.de (M. Selch)

69, 76] and standard references about approximation algorithms [18, 20, 30, 44], branch-and-bound based techniques [7, 75, 78, 79], as well as modern and advanced approaches [16, 26, 82]. Moreover, (deterministic) generalizations with respect to a temporal dimension have recently been proposed in various articles [4, 24, 25]. Given the years of publication, many of the aforementioned articles second the trend displayed in [27, Fig. 1], clearly showing that the BPP still represents an important and widely studied research topic today.

This article deals with an extension of the ordinary BPP to (mutually independent) stochastic item sizes, hereinafter referred to as the *stochastic bin packing problem (SBPP)*.

Remark 1. *Unfortunately, the relevant literature contains co-existing optimization problems that are termed equivalently. One important example, having applications in logistics and supply-chain management [22], is described in [65, 71], where deterministic items have to be assigned to bins under fluctuating profit values or stochastic revenues. Contrary to this interpretation, for the problem studied here, any item size follows a given probability distribution, which is the usual source of uncertainty in stochastic bin packing as also literally supported by the literature reviews in [65, 71]. Hence, no confusion will arise from using the term 'stochastic bin packing problem' throughout this article.*

In this setting, the aim is to assign the items to a minimum number of unit capacity bins while overloading a bin is allowed up to a given tolerable limit. For the sake of completeness, we highlight the fact that, in some publication, this problem is also called the *bin packing problem with chance constraints* (following a similar terminology in the context of knapsack problems [51]), see for instance [21] and references therein. Whichever name is preferred, considering this generalization can be motivated by the leading European research project "HAEC", see [32], dealing with the architecture and pathways towards highly adaptive energy-efficient computing, e.g., by tailoring the amount of active computing devices[1]. However, from a more global perspective, we would like to focus on another, probably more familiar application example, that is server consolidation in data centers[2]. The energy demand of large-scale data centers or server clusters is expected to grow considerably in the next decade, see [2, 52], contributing to up to 13 percent of the worldwide energy consumption in 2030, see [45] for a general overview and Fig. 1 for three possible scenarios.

The main drivers of this development are manifold. As reported in [6], among others, the number of internet users is going to increase by 1.2 billion people from 2017 to 2022, and this is accompanied by an increase of roughly 10.5 billion internet-ready devices. In addition, the importance of cloud computing steadily increases and broadband speed is expected to almost double over the same period. According to this study, these and many other trends lead to an estimated annual increase (more formally, a *compound annual growth rate (CAGR)*) in IP traffic of 26% resulting in a tripling over the entire observation period, see Fig. 2.

Obviously, coping with this huge traffic inevitably requires an enormous number of active servers, most of which typically not being used optimally (for fear of not being able to guarantee high availability during peak times) as independently revealed by several studies [23, 57, 61]. Having in mind the disproportional amount[3] of energy processing units consume when being idle, underutilized or overloaded, see [42] or [83, Fig. 2.2], the previous observation seems even more problematic. Overall, this alarming development of energy consumption has caused deep concerns in industry and scientific communities [13, 48, 56], strongly motivating researchers to move

[1] Another example from the area of communications theory was published in [50]. Therein, the authors address the problem of assigning connections with varying transmission rates (so called *bursty connections*) to possible links.

[2] Applying bin packing approaches to job scheduling problems in computer science has been investigated for many decades, dating back to at least the 1970 years, see [19].

[3] Typically, there is a linear relationship between the power consumption of a server and its CPU utilization level, already generating roughly 70% of the full-load power in the idle state [10, 11, 85].

Figure 1: Three predictions for the energy consumption in terawatt-hours (TWh) of data centers. The original figure has been published in [1, Fig. 4], but a modified version also appeared in [45].

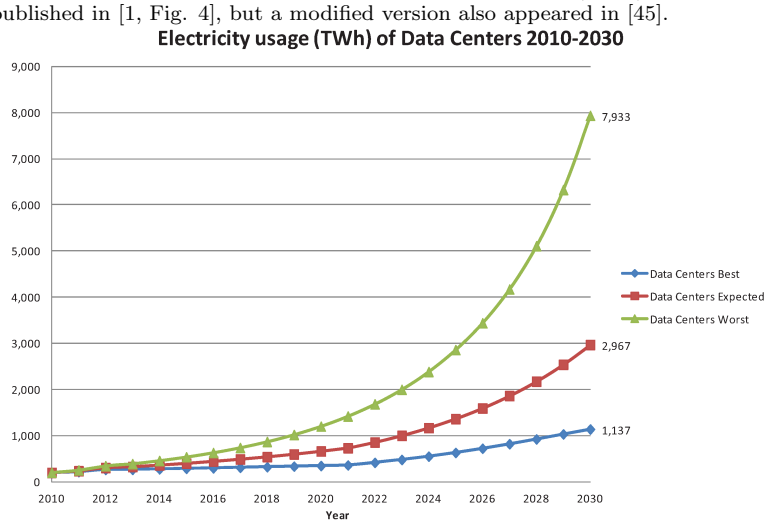
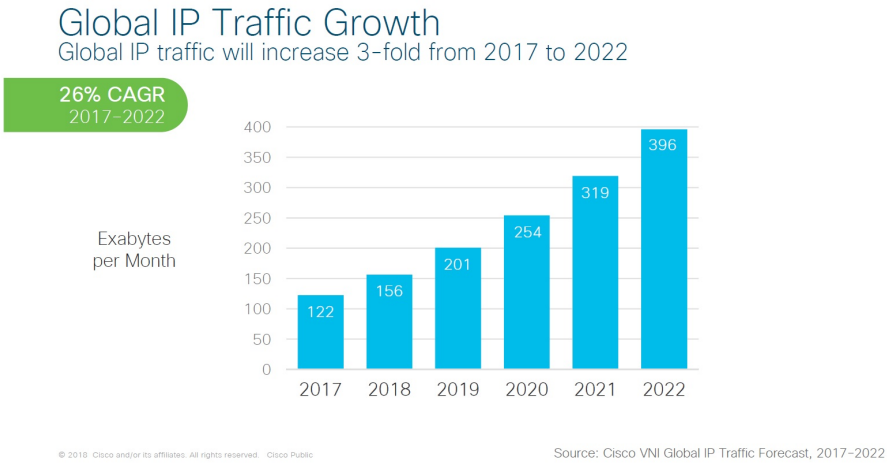


Figure 2: Predicted development of the data center IP traffic according to [6].



towards balancing the supply of and the demand for computing resources as a key issue to obtain energy-efficient server consolidations.

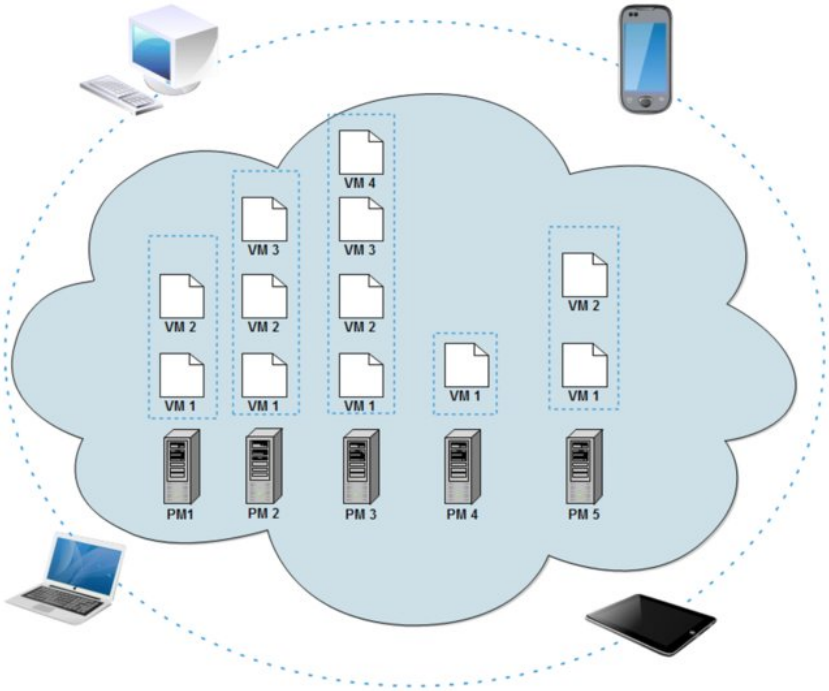
Although numerous approaches have been presented in the literature in recent years, they all share the challenge of predicting future job characteristics as accurately as possible because, in practice, the cloud provider has to select the required resources (for the incoming jobs) in advance, see [21]. To reiterate, the scheduling is of *offline-type* and it has to be performed *before* witnessing the true workloads which in turn are known to mostly be highly volatile, in general, see [8, 46].

Remark 2. *Depending on the particular scenario given by the intended application, stochastic optimization problems can typically be handled as single-stage or two-stage (multi-stage) decision problems, see [12] for a general introduction or [53] for the special case of the closely related stochastic knapsack problem. To avoid misinterpretations, we stress that here a single-stage problem is considered, i.e., the final assignment has to be made before the random parameters are revealed and a subsequent modification of this decision is not allowed. However, for the sake of completeness, one possible way to transform the SBPP to a two-stage decision process will be*

extensively discussed in Remark 7 in Sect. 2, after the required notations have been introduced.

To reasonably address these fluctuations and to better account for the uncertainty of (future) resource demands, treating workloads in a probabilistic way turned out to be a promising approach [42, 62, 80, 84]. Consequently, aiming to reduce the overall energy consumption of a server cluster by assigning stochastic workloads to a minimum number of processing units can be addressed and modeled by an SBPP. Especially in that part of the computer science literature that clearly focuses on applicability, this problem is often equipped with further challenging aspects (such as dynamic scheduling [73], migrations [49], or additional resources [70]) and termed as the *virtual machine placement (VMP) problem*. However, since we are predominantly taking a mathematical approach here, we will not use this terminology (which is partly shown in Fig. 3). Instead, to also get a good overview of the current state of that literature, we recommend the survey articles [54, 72, 74] to the inclined reader.

Figure 3: A simple schematic of virtual machine placement according to [72].



By way of example, as brilliantly summarized in [54, Tab. XVII], a wide variety of different workload distributions has been considered by various articles. Early publications on stochastic bin packing typically dealt with uniformly distributed item sizes, either in a continuous [55, 67, 68] or a discrete setting [17], as well as Bernoulli-type [50] or exponential distributions [39]. In this work, we only consider normally distributed workloads, although the general theoretical approach is not limited to this assumption, see [59]. The reasons for that are threefold:

- From a mathematical point of view, probability distributions being "stable" under convolution are required to avoid dealing with very difficult integrals not having a closed-form solution. Within this group, the normal distribution is the only challenging one, as it leads to a non-linear capacity constraint (for each bin), while other distributions can be transformed to conventional BPP, see [59, Remark 1], or turn out to be rather unsuitable for our purposes (e.g., Bernoulli trials).
- In the relevant literature, considering normally distributed workloads is a common approach [21, 43, 53, 80] or reasonable approximation, see [59, Remark 3].

- From a practical point of view, assuming normally distributed workloads is warrantable for many real-world data, including the instances from [66] we consider in the computational part, see [84, Fig. 4].

In recent years, many efforts[4] have been done to find approximate solutions for the problem under consideration, e.g., based on lower bounds [59] or heuristic approaches [21, 80], partly involving so called "effective item sizes" (meaning that the stochastic sizes are substituted by deterministic ones easier to handle [15]). However, co-locating jobs in such a way that they neither contend for resources unnecessarily (leading to latency in the execution) nor underutilize them considerably, ideally requires an optimal assignment strategy. A first attempt in this direction has been done in [59] where two integer models are presented, the general ideas of which were shown to be competitive, see [42]. Given the fact that these formulations are currently (still) limited to moderately-sized instances, one main challenge in this area is scalability. Moreover, from a theoretical point of view, knowing the exact optimal value is important to accurately evaluate the performance of approximation algorithms. To this end, after having repeated some important definitions and notations in Sect. 2, the present work shall foster theoretical approaches contributing to further increase the size of problems that can be solved in a reasonable amount of time. More precisely, the main achievements (together with the structure and contents of this manuscript) are the following:

- We present two lower bounds both of which are shown to dominate the previously best bound introduced in [59] either in any case or at least for larger instance sizes. Remarkably, one of these bounds is based on a mixed-integer nonlinear relaxation of the SBPP and can be easily obtained by a fractional next fit decreasing algorithm in $\mathcal{O}(n \cdot \log(n))$. ($\rightarrow$ Subsect. 3.1)
- We present two new heuristics having (slightly) better approximation results than the currently best algorithm from the literature, a local search method from [21]. Moreover, compared to this state-of-the-art method having complexity $\mathcal{O}(n^3)$, our approaches can be performed in $\mathcal{O}(n \cdot \log(n))$. ($\rightarrow$ Subsect. 3.2)
- For any of the approximate solutions contained in both of the previous points, the theoretical worst-case performances are extensively discussed. ($\rightarrow$ Sect. 3)
- We consider and apply several well-known linearization techniques to transform the nonlinear integer program into more tractable linear formulations of the SBPP. ($\rightarrow$ Sect. 4).
- All the approaches are compared based on extensive computational results. In particular, our numerical experiments involve both randomly generated and real-world data, thus forming a set of differently characterized test categories. ($\rightarrow$ Sect. 5)

Overall, the results of this work make valuable contributions to the theory and practice of exact and approximate solution strategies for the SBPP, which will hopefully lead to even broader scientific activity in the future in the subject areas addressed here.

2. Preliminaries and Notation

Throughout this article we consider a given number $n \in \mathbb{N}$ of *items* (or *jobs*), indexed by $i \in I := \{1, \ldots, n\}$, that shall be assigned to as few *bins* (or *servers*) of capacity $C \in \mathbb{N}$ as possible. The *size* (or *workload*) c_i of item $i \in I$ is assumed to follow a given probability distribution $\mathcal{P}_i$. As motivated in the introductory part, here we only consider normally distributed workloads, i.e., $c_i \sim \mathcal{P}_i := \mathcal{N}(\mu_i, \sigma_i^2)$, where the parameters μ_i and σ_i^2 refer to the *mean value* and the

[4] To not overload this introductory section, we decided to shift a more detailed description of what precisely has been established in the literature to the beginning of the respective parts of this paper.

variance, respectively, of item $i \in I$. To make the notation more convenient, we will nevertheless use the more abstract symbol $\mathcal{P}_i$ in some places.

Given the fact that the probability density function of a normal distribution is strictly positive on $\mathbb{R}$, the above description obviously leads to two "problems":

(a) On the one hand, we cannot simply demand that the item sizes assigned to the same bin sum up to at most C, as there is always a small probability of exceeding this value.

(b) On the other hand, real workloads cannot be negative, so that (ideally) truncated normal distributions should be used.

These challenges will be addressed in the following way: To cope with (a), we allow the server to be overloaded (in a probabilistic sense) up to a maximal tolerable limit of $\varepsilon > 0$, hereinafter referred to as an *error bound*. As regards (b), in our computations, the probability for negative workloads is considerably small[5], so that using the (ordinary) normal distribution is an almost exact approximation of the true characteristics.

Definition 1. *A tuple* $E = (n, \boldsymbol{c}, C, \boldsymbol{\mathcal{P}}, \varepsilon)$ *with* $\boldsymbol{c} = (c_1, \ldots, c_n)^\top$, $\boldsymbol{\mathcal{P}} = (\mathcal{P}_1, \ldots, \mathcal{P}_n)$, *$n$ items of size* $c_i \sim \mathcal{P}_i$, $i \in I$, *a bin capacity C, and an error bound* $\varepsilon > 0$ *is called an instance of the SBPP.*

Remark 3. *From now on we will use the fact that it is always possible to normalize the bin capacity C (and the workloads c_i) to $C = 1$.*

For any instance E, we can refer to the possible item combinations by binary vectors $\boldsymbol{a} \in \mathbb{B}^n$ modeling whether item $i \in I$ is contained in the bin ($a_i = 1$) or not ($a_i = 0$).

Definition 2. *Let E be an instance of the SBPP. Any vector* $\boldsymbol{a} \in \mathbb{B}^n$ *satisfying*

$$\mathbb{P}\left[\boldsymbol{c}^\top \boldsymbol{a} > 1\right] \leq \varepsilon, \tag{1}$$

the so called stochastic capacity constraint, is called a (feasible) pattern (or (feasible) consolidation), see Fig. 4 for an exemplary illustration. The set of all patterns will be referred to as $P(E)$.

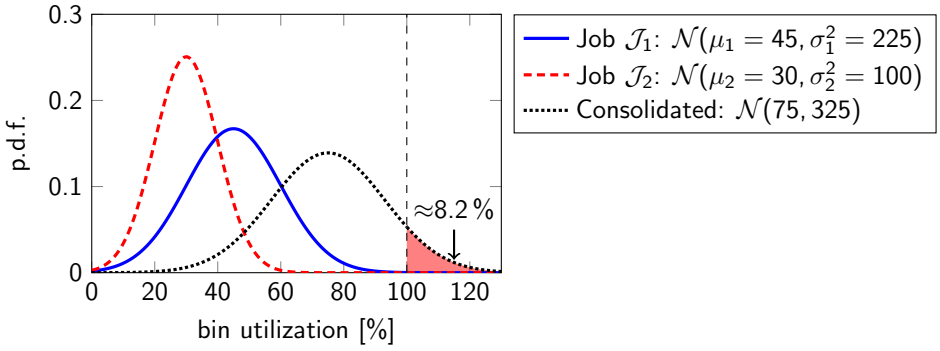


Figure 4: A schematic of an assignment of two stochastic items (illustrated by their *probability density functions* p.d.f.) to one bin. For any $\varepsilon > 0.082$ this assignment is feasible.

[5]For randomly generated data, this can be ensured by drawing the variances σ_i^2 from an interval that depends on the previously selected mean value μ_i, see [59, Sect. 4.3]. Moreover, the real-world examples from a Google data trace [66], we intend to use later in our experiments, are in accordance with this assumption, too.

Remark 4. *For any $i \in I$, we always demand $\mathbb{P}[c_i > 1] \leq \varepsilon$ to ensure the solvability of the SBPP, see also [59, Theorem 1].*

It is a well known result that the sum of normal distributions is again normally distributed, i.e., we have

$$\boldsymbol{c}^\top \boldsymbol{a} \sim \mathcal{N}\left(\sum_{i \in I} \mu_i a_i, \sum_{i \in I} \sigma_i^2 a_i\right),$$

so that Condition (1) can be rephrased to

$$\sum_{i \in I} \mu_i a_i + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2 a_i} \leq 1, \tag{2}$$

where $q_{1-\varepsilon}$ is the $(1-\varepsilon)$-quantile of a standard normal distribution $\mathcal{N}(0,1)$.

Remark 5. *Practically, it is not possible to interpret Inequality (2) as the feasibility condition of an appropriately defined ordinary BPP. Indeed, we have*

$$\sum_{i \in I} \mu_i a_i + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2 a_i} = \sum_{i \in I} \left(\mu_i + \frac{q_{1-\varepsilon}^2 \sigma_i^2}{q_{1-\varepsilon} \cdot \sqrt{\sum_{j \in I} \sigma_j^2 a_j}}\right) \cdot a_i,$$

where the coefficients

$$c_i^{eq} := \mu_i + \frac{q_{1-\varepsilon}^2 \sigma_i^2}{q_{1-\varepsilon} \cdot \sqrt{\sum_{j \in I} \sigma_j^2 a_j}} \tag{3}$$

are usually termed as the equivalent size of item $i \in I$, see [80, Sect. III.B]. As clearly to be seen, the value of c_i^{eq} does not only depend on the data attached to item i, but also on the other items appearing in the considered pattern. Since, a priori, it is not known which items will be combined in an optimal solution, a direct reformulation of Condition (2) as an ordinary knapsack constraint is impossible.

By means of this characterization, a first nonlinear model for the SBPP can be introduced. To this end, let us consider an upper bound $u \in \mathbb{N}$ for the number of bins required[6] to accommodate all items. Then, we define decision variables $y_k \in \mathbb{B}$, $k \in K := \{1, \ldots, u\}$, stating whether bin k is used ($y_k = 1$) or not ($y_k = 0$). Moreover, we use assignment variables $x_{ik} \in \mathbb{B}$, $(i,k) \in I \times K$, to indicate whether item i is packed into bin k ($x_{ik} = 1$) or not ($x_{ik} = 0$). With these ingredients, we obtain the

Nonlinear Assignment Model for the SBPP

$$z = \sum_{k \in K} y_k \to \min$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{4}$$

$$\sum_{i \in I} \mu_i x_{ik} + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2 x_{ik}} \leq 1, \qquad k \in K, \tag{5}$$

$$x_{ik} \leq y_k, \qquad i \in I, k \in K, \tag{6}$$

$$y_k \in \mathbb{B}, \qquad k \in K, \tag{7}$$

[6] For instance, $u = n$ can always be used as an upper bound. More sophisticated choices will be discussed later in Subsect. 3.2.

$$x_{ik} \in \mathbb{B}, \qquad (i,k) \in I \times K. \tag{8}$$

Clearly, the objective function minimizes the number of bins required to pack all items exactly once which is conveyed by (4). Moreover, Conditions (5) ensure that any bin is packed in a feasible manner according to Criterion (2). In this formulation, the coupling between the two types of variables is done by Constraints (6) stating that a bin has to be counted ($y_k = 1$) whenever at least one item is assigned to it. In the opposite direction, packing items to a bin that is not used ($y_k = 0$) cannot happen. In most cases, we will refer to the optimal value of an instance E by $z^\star := z^\star(E)$. However, especially when considering performance ratios, also the common term $OPT(E)$ is applied.

Remark 6. *Note that the coupling of the x- and y-variables could also be manifested within Conditions (5), if y_k would appear on the right hand side. In this case, Constraints (6) would not be required for the integer program. However, they are representing valid inequalities, so that their presence could boost the continuous relaxation. We will return to this point later in Remark 21.*

As the model presented above is nonlinear, its structure is rather unfavorable for common solvers like CPLEX or Gurobi. To deal with this issue, our approach is twofold: In the next section, we investigate lower and upper bounds to approximate the optimal value of the SBPP. Afterwards, in Sect. 4, we present several linearization techniques to transform the given model into a more tractable formulation.

Remark 7. *To conclude this mathematical introductory section, we would like to point out once again that the optimization problem presented here is a one-stage decision. Of course, other practical scenarios can also lead to two-stage decision problems, e.g., if jobs can be removed from overloaded servers afterwards (which, however, causes penalty costs). Based on a similar setting for the closely related stochastic knapsack problem [53], here we would like to exemplarily illustrate how item rejections can be modeled as a second-stage decision. To this end, let us introduce the following parameters:*

- *$r_i > 0$ is the revenue for assigning one unit of item $i \in I$ to some server,*
- *$\alpha > 0$ are the costs for using one server,*
- *$d_i > 0$ are the costs for removing one unit of item $i \in I$ from some server in the second stage.*

Note that we are using completely homogeneous servers, so that we did not consider an additional server-dependent index $k \in K$ in the previous parameters. But we have to assume $d_i > r_i$ for any $i \in I$ so that rejecting an item (in the second stage) produces a negative term in the objective function. In addition, we introduce binary variables $u^-_{ik} \in \mathbb{B}$ to model whether item i is removed from bin k ($u^-_{ik} = 1$) or not ($u^-_{ik} = 0$). Moreover, $1_M[\cdot]$ represents the indicator function of the set $M \subseteq \mathbb{R}$, whereas $\mathbb{E}[\cdot]$ refers to the expected value. Then, the problem can be modeled as:

Two-Stage Stochastic Model (Stage 1)

$$\mathbb{E}\left[\sum_{i\in I}\sum_{k\in K} r_i c_i x_{ik}\right] - \alpha \cdot \mathbb{E}\left[\sum_{k\in K} y_k\right] + \mathbb{E}\left[\mathcal{Q}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{c})\right] \to \max$$

$$\text{s.t.} \quad \sum_{k\in K} x_{ik} = 1, \qquad i \in I,$$

$$\mathbb{P}\left[\sum_{i\in I} c_i x_{ik} > 1\right] \le \varepsilon, \qquad k \in K,$$

$$x_{ik} \le y_k, \qquad i \in I, k \in K,$$

$$y_k \in \mathbb{B}, \qquad k \in K,$$
$$x_{ik} \in \mathbb{B}, \qquad (i,k) \in I \times K,$$

where

Two-Stage Stochastic Model (Stage 2)

$$Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = -\sum_{i \in I} \sum_{k \in K} d_i c_i(\omega) u_{ik}^- \to \max$$

$$\text{s.t.} \qquad u_{ik}^- \leq 1_{(0,\infty)} \left[\sum_{i \in I} c_i(\omega) x_{ik} - 1 \right] \cdot x_{ik}, \qquad i \in I, k \in K, \tag{9}$$

$$\sum_{i \in I} (x_{ik} - u_{ik}^-) c_i(\omega) \leq y_k, \qquad k \in K, \tag{10}$$

$$u_{ik}^- \in \mathbb{B}, \qquad i \in I, k \in K, \tag{11}$$

Overall, the objective is to maximize the expected profit consisting of the total item revenues and the total costs for using servers and rejecting items (in the second stage). Apart from the objective function, the first-stage model corresponds to the nonlinear assignment model introduced before, so here we only focus on the second-stage decision. Conditions (9) are nonlinear restrictions modeling that item $i \in I$ can only be removed from server $k \in K$, if it has been assigned to it in the first stage (i.e., $x_{ik} = 1$) and if the server is overloaded (after the true workloads $c_i(\omega)$ have been revealed), meaning that $\sum_{i \in I} c_i(\omega) x_{ik} - 1 \in (0, \infty)$ has to hold. In any other scenario, u_{ik}^- is set to zero automatically. Constraints (10) make sure that (after having possibly rejected some items) the server capacity is respected for any $k \in K$. We would like to underline again that, depending on the practical scenario, other second-stage decisions can also be meaningful. By way of example, more servers could be added to accommodate possibly rejected items. However, as literally stated in [53, Sect. 1], the exact evaluation of the objective function becomes very difficult, or even impossible, if there are infinitely many scenarios with non-zero probability. Hence, as we just intend to present a rough outlook to the field of two-stage stochastic programming, these further aspects shall not be discussed within this article. Instead, we refer the interested reader to the book [12], containing lots of detailed examples related to the general topic.

To conclude this remark, let us briefly mention that we have stated the first-stage objective function in a very general and abstract way following the conventions of stochastic optimization. However, this can be simplified due to the following observations:

$$\mathbb{E}\left[\sum_{i \in I} \sum_{k \in K} r_i c_i x_{ik} \right] = \mathbb{E}\left[\sum_{i \in I} r_i c_i \sum_{k \in K} x_{ik} \right] \overset{(4)}{=} \mathbb{E}\left[\sum_{i \in I} r_i c_i \right] = \sum_{i \in I} r_i \mathbb{E}[c_i] = \sum_{i \in I} r_i \mu_i,$$

$$\mathbb{E}\left[\sum_{k \in K} y_k \right] = \sum_{k \in K} y_k.$$

Observe that the latter holds because the y-variables are deterministic. Altogether, the first-stage objective function can also be formulated as

$$\sum_{i \in I} r_i \mu_i - \alpha \sum_{k \in K} y_k + \mathbb{E}\left[\mathcal{Q}(x, y, c) \right],$$

so without a second stage decision, we would end up with the first two terms only. But maximizing $\sum_{i \in I} r_i \mu_i - \alpha \sum_{k \in K} y_k$ is the same as minimizing $\sum_{k \in K} y_k$ (in terms of optimality, not in terms of the precise objective value), leading to the objective function considered throughout this article.

3. Lower and Upper Bounds

3.1. Lower Bounds

In [59, Sect. 4.1], two lower bounds for the SBPP have already been introduced. For a given instance E, the first bound requires an upper bound $\gamma_0 := \gamma_0(E) \in \mathbb{N}$ for the maximum number of items appearing in a feasible pattern $a \in P(E)$, and it is defined by

$$lb_0 := lb_0(E) := \left\lceil \frac{n}{\gamma_0} \right\rceil .$$

As the computation of γ_0 is based on solving a knapsack problem, from a theoretical point of view determining this bound has to be considered as rather difficult. However, for the instance sizes we consider, the $\mathcal{NP}$-hardness of the knapsack problem does not matter within the calculations.

For $\varepsilon \in (0, 0.5]$ the second lower bound is given by

$$lb_1 := lb_1(E) := \left\lceil \sum_{i \in I} \mu_i + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2} \right\rceil \tag{12}$$

and can be computed in $\mathcal{O}(1)$, see [59, Lemma 4]. For the sake of a better comprehensibility, we briefly repeat that it can be obtained by summing up Conditions (5) for all bins of an optimal solution in the following way:

$$\begin{aligned} z^\star &\geq \sum_{k \in K} \left(\sum_{i \in I} \mu_i x_{ik} + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2 x_{ik}} \right) = \sum_{i \in I} \mu_i \sum_{k \in K} x_{ik} + q_{1-\varepsilon} \sum_{k \in K} \sqrt{\sum_{i \in I} \sigma_i^2 x_{ik}} \\ &\geq \sum_{i \in I} \mu_i \sum_{k \in K} x_{ik} + q_{1-\varepsilon} \sqrt{\sum_{i \in I} \sigma_i^2 \sum_{k \in K} x_{ik}} \overset{(5)}{=} \sum_{i \in I} \mu_i + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} \sigma_i^2}. \end{aligned}$$

Although it could theoretically be demonstrated that there is no dominance relation between these two bounds, see [59, Remark 5], the second one (lb_1) performed very much better in the computational tests so that, empirically, not a single instance with $lb_0 \geq lb_1$ was found. Hence, for the current investigation, we consider lb_1 as the state of the art and do not include lb_0 anymore.

According to the numerical experiments from [59], the bound lb_1 performed very well for instances up to $n = 100$ items, often leading to reasonable approximations of the optimal value. However, the trend also showed that its performance became a bit worse when n increases. Hence, for larger instance sizes, the very good approximation quality cannot be guaranteed. To this end, here we present two new lower bounds, the first of which (also representing a combinatorial bound) will be shown to dominate lb_1 especially for larger instances. The second lower bound is more complex to calculate as it requires to solve a particular relaxation of the nonlinear model presented in Sect. 2. However, it can be shown that its optimal value is always performing better than lb_1.

Based on the observations in Remark 5, we obtain the following result.

Lemma 8. *Let E be an instance of the SBPP, then*

$$lb_2 := lb_2(E) := \left\lceil \sum_{i \in I} \left(\mu_i + \frac{q_{1-\varepsilon}^2 \sigma_i^2}{1 - \mu_i} \right) \right\rceil \tag{13}$$

represents a lower bound for the optimal value (of E).

Proof. Let $z^\star$ denote the optimal value of E, and let $x^\star$ represent the corresponding assignment variables in an optimal solution. According to (4), for any item $i \in I$, there is a unique bin $k(i) \in K$ with $x^\star_{ik(i)} = 1$, so that Condition (5) leads to

$$q_{1-\varepsilon}\sqrt{\sum_{j\in I}\sigma_j^2 x^\star_{jk(i)}} \leq 1 - \sum_{j\in I}\mu_j x^\star_{jk(i)} \leq 1 - \mu_i x^\star_{ik(i)} \leq 1 - \mu_i. \tag{14}$$

Now, using the idea of equivalent item sizes from (3), we obtain

$$1 \geq \sum_{i\in I}\mu_i x^\star_{ik} + q_{1-\varepsilon}\sqrt{\sum_{i\in I}\sigma_i^2 x^\star_{ik}} = \sum_{i\in I}\left(\mu_i + \frac{q^2_{1-\varepsilon}\sigma_i^2}{q_{1-\varepsilon}\sqrt{\sum_{j\in I}\sigma_j^2 x^\star_{jk}}}\right) x^\star_{ik}$$

for any bin $k \in K$. A summation over all $k \in K$ then finally leads to

$$\begin{aligned} z^\star &\geq \sum_{k\in K}\left(\sum_{i\in I}\left(\mu_i + \frac{q^2_{1-\varepsilon}\sigma_i^2}{q_{1-\varepsilon}\sqrt{\sum_{j\in I}\sigma_j^2 x^\star_{jk}}}\right) x^\star_{ik}\right) = \sum_{i\in I}\sum_{k\in K}\left(\mu_i + \frac{q^2_{1-\varepsilon}\sigma_i^2}{q_{1-\varepsilon}\sqrt{\sum_{j\in I}\sigma_j^2 x^\star_{jk}}}\right) x^\star_{ik} \\ &= \sum_{i\in I}\left(\mu_i + \frac{q^2_{1-\varepsilon}\sigma_i^2}{q_{1-\varepsilon}\sqrt{\sum_{j\in I}\sigma_j^2 x^\star_{jk(i)}}}\right) \overset{(14)}{\geq} \sum_{i\in I}\left(\mu_i + \frac{q^2_{1-\varepsilon}\sigma_i^2}{1-\mu_i}\right) \end{aligned}$$

which proves the claim after rounding-up the right hand side. □

Observe that this bound is valid for any choice of $\varepsilon \in (0, 1)$, because $q_{1-\varepsilon}$ only appears in a squared term. More importantly, the true merit of this new bound is that it dominates lb_1 whenever a specific condition holds.

Lemma 9. *Let E be an instance of the SBPP. If $\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2 \geq 1$ is true, then we have $lb_2 \geq lb_1$.*

Proof. Due to the given property, we surely have

$$\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2 \geq \sqrt{\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2}$$

which further leads to

$$\sum_{i\in I}\frac{q^2_{1-\varepsilon}\sigma_i^2}{1-\mu_i} \geq \sqrt{\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2}.$$

By adding $\sum_{i\in I}\mu_i$ on both sides of this inequality, the latter directly leads to $lb_2 \geq lb_1$ and we are done. □

Although we only verified that $lb_2 \geq lb_1$ has to hold, the proof also shows that equality can only happen when $\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2$ is rather close to one. In particular, having $\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2 \to \infty$, the difference between both bounds can become arbitrarily large.

Remark 10. *Especially for randomly generated instances the parameters of which are drawn according to a uniform distribution on a fixed interval, the condition $\sum_{i\in I} q^2_{1-\varepsilon}\sigma_i^2 > 1$ is (on average) more likely to happen if n is large. Hence, for increasing values of n, we should expect that lb_2 strictly beats lb_1 in most of the cases.*

For $\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2 < 1$, we cannot simply use the argumentation from the above proof to state an opposite relation between both lower bounds. Actually, in this setting no general dominance can be observed. Indeed, having $\mu_i = \mu$ for all $i \in I$, we conclude that

$$\left(\sum_{i\in I}\mu_i + \sum_{i\in I}\frac{q_{1-\varepsilon}^2\sigma_i^2}{1-\mu}\right) - \left(\sum_{i\in I}\mu_i + \sqrt{\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2}\right) = \frac{\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2}{1-\mu}\cdot\left(1 - \frac{1-\mu}{\sqrt{\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2}}\right),$$

so that both relations are possible depending on whether $\sqrt{\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2} > 1-\mu$ (that is, $lb_2 \geq lb_1$) or $\sqrt{\sum_{i\in I} q_{1-\varepsilon}^2\sigma_i^2} < 1-\mu$ (that is, $lb_2 \leq lb_1$) holds. Whichever the case may be, again, the absolute difference (between these bounds) can become arbitrarily large.

As a second step, we intend to present a lower bound of even better quality. More precisely, this bound will be shown to always dominate the previous bounds lb_1 and lb_2. However, its calculation is based on an algorithm and the theoretical arguments are very technical and difficult. To this end, here we only present the key ideas, whereas the full justification is shifted to a very extended appendix. For a given instance E, we consider the optimization problem $\Phi(E)$, defined by

Relaxation of the Nonlinear Assignment Model

$$z = \sum_{k\in K} y_k \to \min$$

$$\text{s.t.} \quad \sum_{k\in K} x_{ik} = 1, \qquad i \in I, \tag{15}$$

$$\sum_{i\in I} x_{ik}\mu_i + q_{1-\varepsilon}\cdot\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2} \leq 1, \qquad k \in K, \tag{16}$$

$$x_{ik} \leq y_k, \qquad i \in I, k \in K, \tag{17}$$

$$x_{ik} \geq 0, \qquad i \in I, k \in K, \tag{18}$$

$$y_k \in \{0,1\}, \qquad k \in K. \tag{19}$$

Clearly, this is a relaxation of the nonlinear assignment model for the SBPP, thus resulting in a lower bound, hereinafter referred to as $lb_3 := lb_3(E)$.

Remark 11. *Similar to the proofs establishing the lower bounds for the integer problem it can be shown that lb_1 and lb_2 are also lower bounds for the optimal value of the system $\Phi(E)$, thus leading to $lb_3 \geq \max\{lb_1, lb_2\}$. Among the three bounds presented, lb_3 should thus be expected to perform best.*

However, $\Phi(E)$ is still a mixed-integer nonlinear optimization problem, typically making its solution very costly. Fortunately, in this special case, a tailored next fit decreasing heuristic (see Alg. 1) can solve $\Phi(E)$ to optimality.

Theorem 12. *For $\varepsilon \in (0, 0.5)$, the system $\Phi(E)$ can be solved by Alg. 1 in $\mathcal{O}(n\cdot\log(n))$.*

Proof. Given the theoretical background extensively discussed in the appendix, we only have to prove the complexity of the algorithm. To this end, note that the sorting can be performed by MergeSort in $\mathcal{O}(n\cdot\log(n))$, for instance. Moreover, the Next Fit algorithm itself has a complexity of $\mathcal{O}(n)$, especially since the computation of the fraction $x_{ik} \in [0,1]$ leads to a quadratic equation, which can be solved in $\mathcal{O}(1)$. Hence, the claim is proved. □

Remark 13. *Most likely, the lower bound is also valid for $\varepsilon = 0.5$. However, the proof presented in the appendix cannot cope with that value since $q_{1-\varepsilon} > 0$ is explicitly required. Typically, this large error bound is not used at all in practical scenarios, so that we do not intend to investigate this case in more details.*

Algorithm 1 Fractional Next Fit Decreasing

Input: Instance E of the SBPP.

Sort the items in non-increasing order of σ_i^2/μ_i.
Open a first bin $k = 1$ and set $i = 1$.
while $i \leq n$ **do**
  Compute the maximum fraction $x_{ik} \in [0, 1]$ of item i that can be assigned to bin k.
  if $x_{ik} = 1$ **then**
    Assign the complete item i to bin k. Set $i := i + 1$.
  else
    Assign the fraction $x_{ik} < 1$ of item i to bin k. Create a new bin and assign the remaining portion $1 - x_{ik}$ to it. Set $k := k + 1$ and $i := i + 1$.
  end if
end while

Output: k

We strongly emphasize that Alg. 1 does not have to lead to an optimal solution of $\Phi(E)$, if an arbitrary item order is chosen.

Example 1. *We consider the instance given by $n = 15$, $\varepsilon = 0.05$, and the item characteristics listed in Tab. 1.*

Table 1: Input data of an instance underlining the importance of the correct item order (The row 'pos' indicates the position of the object in a list, which is ordered with respect to non-increasing values σ_i^2/μ_i.)

i	1	2	3	4	5	6	7	8
μ_i	0.200	0.150	0.103	0.130	0.155	0.070	0.210	0.030
σ_i^2	0.015	0.010	0.015	0.005	0.005	0.003	0.009	0.005
pos	7	9	4	12	14	11	10	3

i	9	10	11	12	13	14	15
μ_i	0.140	0.060	0.070	0.179	0.090	0.171	0.010
σ_i^2	0.020	0.004	0.002	0.017	0.003	0.030	0.002
pos	5	8	15	6	13	2	1

For these data, we obtain $z^\star = 3$ and $lb_3 = 3$, where the latter is determined by the following nonzero assignment variables (listed in the order they are chosen in Alg. 1)

- *bin $k = 1$: $x_{15,1} = x_{14,1} = x_{8,1} = x_{3,1} = x_{9,1} = 1$, $x_{12,1} \approx 0.455$,*
- *bin $k = 2$: $x_{12,2} \approx 0.545$, $x_{1,2} = x_{10,2} = x_{2,2} = 1$, $x_{7,2} \approx 0.693$,*
- *bin $k = 3$: $x_{7,3} \approx 0.307$, $x_{6,3} = x_{4,3} = x_{13,3} = x_{5,3} = x_{11,3} = 1$.*

On the other hand, a simple fractional next fit algorithm without any pre-sorting leads to the value $z = 4$. Consequently, we see that the item order is not only important to generate an optimal solution of $\Phi(E)$, but also to guarantee that a lower bound for $z^\star$ is obtained by the current approach.

To theoretically evaluate the performance of lb_3, we state the following result:

Lemma 14. *Let E be an instance of the SBPP, then we have $OPT(E) \leq 2 \cdot lb_3 - 1$. More precisely, also $\sup_E \frac{OPT(E)}{lb_3(E)} = 2$ holds.*

Proof. Let us consider the fractional optimal solution obtained by Alg. 1. Observe that almost all values x_{ik} in this solution are integer, except for those constellations where an item i was split into two successive bins k and $k+1$ (like in the previous example). As lb_3 denotes the optimal value of $\Phi(E)$, splitting an item can happen at most $lb_3 - 1$ times. If we recombine any such item and assign it to an additional bin, then at most $lb_3 - 1$ extra bins will be required. Hence, the optimal solution of $\Phi(E)$ can be transformed into a feasible solution of the SBPP having at most $2 \cdot lb_3 - 1$ bins, which proves the first claim.

Let us now consider an instance with $n \geq 2$ items having $\mu_i = 1/2 + 1/n$ and $\sigma_i^2 = 0$ for all $i \in I$. An optimal solution of the SBPP obviously requires $z^\star = n$ bins, whereas we also observe

$$lb_3 = \left\lceil n \cdot \left(\frac{1}{2} + \frac{1}{n} \right) \right\rceil = \left\lceil \frac{n}{2} + 1 \right\rceil,$$

so that $z^\star / lb_3 \to 2$ holds when n tends to infinity. □

As we will see in the computational results, the empirical performance of this bound is actually much better.

Remark 15. *Since lb_3 was previously shown to dominate lb_1 and lb_2, their respective worst-case performance ratios are now known to be bounded below by the constant 2. However, a more detailed study on these values is not intended to be part of the paper since, in the final simulations (involving the exact approaches), we will always use the best bound lb_3 as a reference value.*

3.2. Upper Bounds

After having considered a set of lower bounds, we now focus on upper bounds for the SBPP. Normally, these bounds are obtained from a heuristic together with a feasible solution. Consequently, also the approximation guarantee of this feasible point is an important theoretical research topic. In this regard, with $ALG(E)$ (resp. $OPT(E)$) denoting the heuristic (resp. optimal) value for a given instance E, the following key results have been obtained in the relevant literature:

- In [15], the authors propose to replace the stochastic item size by a deterministic one, which is defined by $c_i^{det} := 1/N_i$ with

  $$N_i := \max\{N \in \mathbb{N} \,:\, N\mu_i + q_{1-\varepsilon}\sqrt{N}\sigma_i \leq 1\}.$$

  Effectively, the number N_i states how many identical copies of item $i \in I$ would fit into one bin. However, the authors also show that $\sum_{i \in I} c_i^{det} a_i \leq 1$ does not necessarily imply the feasibility condition (2) of the SBPP. To be more precise, only in the case, where $N_i = 1$ holds (i.e., one relatively large item is exclusively contained in the bin), an exact reformulation of (2) is obtained. Altogether, this method can only be applied if the capacity in the deterministic auxiliary problem is drastically reduced[7].

- In [80] a *group packing algorithm* is proposed. More precisely, the items are first divided into various groups, each of which is then assigned to a separate set of bins according to a next fit heuristic. Let $\psi \in (0, 1)$ be fixed; then for $\mu_i \in (\psi, 1)$, $i \in I$, not arbitrarily close to zero and $m \leq n$ denoting the number of different mean values, the performance result

  $$ALG(E) < \left(\sqrt{2} + 1\right) \cdot OPT(E) + m \cdot \left\lfloor \frac{1}{\psi} \right\rfloor$$

[7] After having assigned each item with $N_i = 1$ to a separate bin, it can be shown that $C^{det} = 1/2$ has to be used as the capacity to also ensure feasible patterns for the stochastic problem.

can be obtained, see [80, Theorem 1]. Besides having the rather large (constant) approximation factor $\sqrt{2}+1$, the right hand side of the inequality further depends on the lower bound ψ of the mean values. In particular, for general instances (where only $\mu_i \in (0,1)$ is known), this result cannot be applied.

- In [59], the authors propose a first fit decreasing (FFD) heuristic where the items are sorted according to non-increasing mean values. Finally, in [59, Theorem 3], it is stated that

$$ALG(E) < \frac{5}{2} \cdot OPT(E) + 2\beta \cdot q_{1-\varepsilon}$$

holds, closing the gap that arbitrary instances can also be dealt with. However, also here a relatively large approximation factor (5/2) is part of the formula. Moreover, deriving this statement requires the mean values and the variances to be coupled by some parameter $\beta > 0$ in a way that $\sigma_i \leq \beta \cdot \mu_i$ holds for all $i \in I$. Since we would like to keep the probability of negative item sizes very low, at least $\beta \leq 1/3$ (so that positivity is ensured with more than 99% probability) can be assumed for our purposes. For common choices of ε, the corresponding quantile is always bounded by $q_{1-\varepsilon} < 4$, so that the second term in the above inequality cannot become arbitrarily large as in the previous consideration.

- Another contribution of [59] is based on the fact that $c_i^{eq} \leq \mu_i + q_{1-\varepsilon}\sigma_i$, $i \in I$, holds for the effective item sizes presented in Remark 5. Hence, solving an ordinary BPP with deterministic item sizes $\mu_i + q_{1-\varepsilon}\sigma_i$, $i \in I$, leads to an upper bound for the optimal value of the SBPP. In [59], it is noticed that (from a theoretical point of view) there is no dominance between this approach and the FFD heuristic described in the previous point. However, the FFD algorithm performed slightly better for the instances considered in that article. Moreover, FFD does not require to solve a discrete optimization problem to optimality, so that it is more appropriate also in terms of the run times.

- Recently, in [21, Sect. 5.2] an ordinary first fit heuristic was shown to lead to

$$ALG(E) \leq \frac{9}{4} \cdot OPT(E) + 1,$$

improving on the previously best constant factor and not requiring any further data (as, for instance, the parameters β or $q_{1-\varepsilon}$). Moreover, no sorting is needed so that this algorithm can also be applied in an online scenario. For offline considerations, a rather complex local search algorithm (having complexity $\mathcal{O}(n^3)$ and more than half a page of verbal description) appearing in [21, Sect. 6] actually provides

$$ALG(E) \leq 2 \cdot OPT(E) + 11,$$

the currently best approximation guarantee known in the literature.

While for small instances adding 11 more bins can effectively result in a larger approximation factor[8], for larger instances the cubic complexity can lead to high computation times (compared to standard heuristics like next fit, best fit, first fit, or variants with pre-sorted items). To this end, we suggest two new heuristic strategies (also representing a factor 2 approximation), whose theoretical advantages (compared to the local search approach proposed in [21]) are twofold:

- The relatively large constant term 11 can be reduced to -1 which is particularly interesting for smaller instance sizes.

- The complexity of the algorithm can be reduced considerably from $\mathcal{O}(n^3)$ to $\mathcal{O}(n \cdot \log(n))$. By doing so, also the description of the algorithm becomes much more economic.

[8] By way of example, let us consider instances with $z^\star \leq 11$. Then, we basically have $ALG(E) \leq 3 \cdot OPT(E)$.

Let us first consider the procedure given by Alg. 2.

Algorithm 2 Fractional NFD with Post Processing

Input: Instance E of the SBPP.

1: Apply the Fractional NFD heuristic described in Alg. 1. Let z be the number of (partially) filled bins.
2: Set $k := 1$.
3: **while** $k < z$ **do**
4: Check whether there is an item $i \in I$ with $x_{ik} \in (0,1)$ and $x_{i,k+1} \in (0,1)$. If so, then recombine these two fractions and assign the complete item i to a newly opened bin. Set $k := k + 1$.
5: **end while**

Output: number of partially filled bins

Then, we can state:

Theorem 16. *Let E be an instance of the SBPP, then*

$$ALG(E) \leq 2 \cdot OPT(E) - 1$$

holds, where $ALG(E)$ is the value obtained by Alg. 2. Moreover, this algorithm terminates after at most $\mathcal{O}(n \cdot \log(n))$ steps.

Proof. First of all, note that the complexity of the algorithm is clear. As observed in the proof of Lemma 14, the method described in Alg. 2 generates a feasible solution of the SBPP having at most $2 \cdot lb_3 - 1$ partially filled bins. Together with $lb_3 \leq OPT(E)$ we can conclude

$$ALG(E) \leq 2 \cdot lb_3 - 1 \leq 2 \cdot OPT(E) - 1.$$

□

Obviously, to probably achieve even better results, we should first collect all the recombined items from Alg. 2, and then try to apply a more sophisticated post-processing strategy, e.g., by assigning all the gathered items according to an FFD heuristic. Hence, instead of the naive method from Alg. 2, we only consider the refined strategy illustrated in Alg. 3 within our computational tests. Of course, similarly to the proof of Theorem 16, also this method can be shown to possess a worst-case performance ratio of 2.

Algorithm 3 Fractional NFD with FFD Post Processing

Input: Instance E of the SBPP.

1: Apply the Fractional NFD heuristic described in Alg. 1. Let z be the number of (partially) filled bins.
2: Set $k := 1$.
3: **while** $k < z$ **do**
4: Check whether there is an item $i \in I$ with $x_{ik} \in (0,1)$ and $x_{i,k+1} \in (0,1)$. If so, then recombine these two fractions and store the item. Set $k := k + 1$.
5: **end while**
6: Assign all stored items to the existing bins according to an FFD approach (the pre-sorting criterion is the same as for the Fractional NFD algorithm), i.e., any item is packed into the first bin that is able to accommodate it, and new bins are opened only in the case, where no such bin exists.

Output: number of partially filled bins

Since, in [59], the general suitability of an FFD approach could already be verified, another approach that shall be tested here can be described as follows:

Algorithm 4 First Fit Decreasing with Specific Sorting

Input: Instance E of the SBPP.

1: Initialize an empty bin, and sort all items according to non-increasing values of σ_i^2/μ_i.
2: **for all** $i \in I$ **do**
3: Find the lowest-indexed bin that is able to accommodate item i without violating the feasibility condition. If such a bin does not exist, generate a new (empty) bin and assign item i to it.
4: **end for**

Output: number of partially filled bins

As regards the approximation guarantee, the procedure described in Alg. 4 is equivalent to Alg. 3.

Theorem 17. *Let E be an instance of the SBPP, then we have $ALG(E) \leq 2 \cdot OPT(E) - 1$, where $ALG(E)$ represents the value obtained by Alg. 4. Moreover, the algorithm terminates after $\mathcal{O}(n \cdot \log(n))$ steps.*

Proof. First of all, note that the complexity result is clear. Let us now consider an instance $\widetilde{E}$, where some items of E are "shortened" in a sense that the parameter values μ_i and σ_i^2 of some specific items $i \in I$ (of E) are reduced. To this end, let us study the FFD approach described by Alg. 4 step by step. During the execution of this method, we start with one empty bin and fill it with the arriving items until one of the two cases happens

(a) The bin is completely filled without any remaining capacity. Then, we simply move on to the next (empty) bin.

(b) The bin is filled, but there is some space left, i.e., the remaining capacity is not sufficient to accept the entire next item (say $i^\star \in I$).

While Case (a) does not require any modifications of the input data (of E), we have to consider Case (b) in more details: Let us reduce the parameters of $i^\star$ by redefining $\widetilde{\mu}_{i^\star} := p \cdot \mu_{i^\star}$ and $\widetilde{\sigma}_i^2 := p \cdot \sigma_i^2$, where

$$p := \max \left\{ t \in [0,1] \, : \, \sum_{i<i^\star} \mu_i + t \cdot \mu_{i^\star} + q_{1-\varepsilon} \cdot \sqrt{\sum_{i<i^\star} \sigma_i^2 + t \cdot \sigma_{i^\star}^2} \leq 1 \right\}. \tag{20}$$

Obviously, reducing the parameters in the above way leads to a situation where the first bin is filled exactly by the items $1 \leq i \leq i^\star$. Now, the FFD approach has to open a new bin and assigns the following items until one of the above cases appears. If, in Case(b), a specific item would not fit completely into this second bin, then, again, we reduce the characteristics of this item (by a factor similar to (20)), so that it perfectly fits into the second bin, and so on. At the end, we notice that Alg. 4 applied to $\widetilde{E}$ only generated perfectly filled bins (possibly except for the very last one). Moreover, since the original sorting does not change when both, μ_i and σ_i^2, are multiplied by the same constant p, we know that for $\widetilde{E}$ Alg. 4 actually mimics the Fractional NFD algorithm from Alg. 1 (but here without splitting any item into two parts!), so that an optimal integer solution is obtained. Concluding this first part of the proof, we can particularly state $ALG(\widetilde{E}) = OPT(\widetilde{E})$.

These observations can be applied to obtain the desired inequality given in the statement of the current theorem. Observe that, in the FFD approach for the original instance E, we can at most have one additional bin per "shortened" item, i.e., $ALG(E) \leq 2 \cdot ALG(\widetilde{E}) - 1$ has to hold, since we had to modify the characteristics of at most $ALG(\widetilde{E}) - 1$ items to construct $\widetilde{E}$. All the observed relationships between the heuristic and the optimal values can now be concatenated in the following way

$$ALG(E) \leq 2 \cdot ALG(\widetilde{E}) - 1 = 2 \cdot OPT(\widetilde{E}) - 1 \leq 2 \cdot OPT(E) - 1,$$

where the latter holds due to the fact that $\widetilde{E}$ basically has the same items as E, but some of them are (possibly) reduced in the above sense. Consequently, the claim is proved. □

Remark 18. *Recapitulating the arguments applied in the previous proof, it can be noticed that the same approximation guarantee can also be observed if Alg. 4 would be based on a next fit decreasing or best fit decreasing (BFD) strategy applied to the same pre-sorted item list. As, for the ordinary BPP, it is known that the (asymptotic) worst case performances of FFD and BFD are equal, here we only concentrate on one representative, namely Alg. 4.*

Altogether, we succeeded in proposing two heuristic approaches both of which showing theoretical advantages compared to the currently best formulation (the local search method from [21]) known in the literature. These observations strongly suggest that Alg. 3 and Alg. 4 can be competitive also within the numerical experiments.

4. Exact Solution Approaches

4.1. A Quadratically Constrained ILP from the Literature

In the previous section, we not only dealt with lower and upper bounds, but also stated a first exact approach for the SBPP which is based on the well known Kantorovich model, see [47]. The main drawback of this integer formulation consisted in imposing the stochastic capacity constraint in a nonlinear way making it rather difficult to solve. To this end, we first recapitulate the result from [59, Theorem 2] offering a more tractable feasibility condition.

Theorem 19. *Let an instance E with $0 < \varepsilon \leq 0.5$ be given. Then, a vector $\boldsymbol{a} \in \mathbb{B}^n$ represents a pattern if and only if*

$$\sum_{i \in I} \left(q_{1-\varepsilon}^2 \cdot \sigma_i^2 + 2\mu_i - \mu_i^2 \right) a_i - 2 \sum_{i \in I} \sum_{j > i} \mu_i \mu_j a_i a_j \leq 1, \tag{21}$$

$$\sum_{i \in I} \mu_i a_i \leq 1, \tag{22}$$

hold.

Indeed, contrary to the characterization presented in (2), here a feasible pattern is defined by one linear and one quadratic constraint, so that the SBPP can be written as a *quadratically constrained integer linear program (QCILP)*. Introducing $\alpha_i := q_{1-\varepsilon}^2 \cdot \sigma_i^2 + 2\mu_i - \mu_i^2$, $i \in I$, to abbreviate the coefficients appearing in (21), we obtain the

Quadratically Constrained ILP for the SBPP (Basic Version)

$$z = \sum_{k \in K} y_k \to \min$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{23}$$

$$\sum_{i \in I} \alpha_i x_{ik} - 2 \sum_{i \in I} \sum_{j > i} \mu_i \mu_j x_{ik} x_{jk} \leq y_k, \qquad k \in K, \tag{24}$$

$$\sum_{i \in I} \mu_i x_{ik} \leq y_k, \qquad k \in K, \tag{25}$$

$$y_k \in \mathbb{B}, \qquad k \in K, \tag{26}$$

$$x_{ik} \in \mathbb{B}, \qquad (i,k) \in I \times K. \tag{27}$$

Note that, in this version from the literature, one of the slightly different approaches (already mentioned in Remark 6) to couple the two types of variables is applied. More precisely, instead

of demanding $x_{ik} \leq y_k$ for all $i \in I$ and $k \in K$, the above formulation simply attached the y-variables to the right hand sides of the feasibility conditions. However, in our computational tests, we also briefly discuss whether adding these valid inequalities (of type $x_{ik} \leq y_k$) has any positive effects on the above (and the upcoming) formulation.

In [59], some improvements have been discussed to (i) reduce the number of variables and (ii) avoid symmetric solutions originating from permutations. To this end, the following modifications are performed:

- We (implicitly) renumber the bins so that item $i = 1$ is contained in bin $k = 1$, item $i = 2$ either appears in bin $k = 1$ or in a new bin $k = 2$, and so on. Hence, we only need to consider index combinations $(i, k) \in I \times K$ with $i \leq k$, all of which will be collected in the set $\Delta \subset I \times K$. For a convenient mathematical description of Condition (24), we further require the index sets $\Delta_{ik} := \{j \in I : (j, k) \in \Delta,\, j > i\}$ for all $(i, k) \in \Delta$.
- Given a lower bound $\eta \in \mathbb{N}$, see Subsect. 3.1, it is possible to already set $y_1 = \ldots = y_\eta = 1$, since we know that at least the first η bins will be used in a optimal solution.

These observations finally lead to the

Quadratically Constrained ILP for the SBPP

$$z = \sum_{k \in K} y_k \to \min$$

$$\text{s.t.} \quad \sum_{(i,k) \in \Delta} x_{ik} = 1, \qquad i \in I, \tag{28}$$

$$\sum_{(i,k) \in \Delta} \alpha_i x_{ik} - 2 \sum_{(i,k) \in \Delta} \sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{ik} x_{jk} \leq y_k, \qquad k \in K, \tag{29}$$

$$\sum_{(i,k) \in \Delta} \mu_i x_{ik} \leq y_k, \qquad k \in K, \tag{30}$$

$$\sum_{k=1}^{\eta} y_k = \eta, \tag{31}$$

$$y_k \in \mathbb{B}, \qquad k \in K, \tag{32}$$

$$x_{ik} \in \mathbb{B}, \qquad (i, k) \in \Delta. \tag{33}$$

For simplicity, we did not change the name of the above model obtained by the reduction methods since, from now on, we only refer to the improved version as the state of the art.

4.2. Linearized Formulations

In general, integer minimization problems involving non-convex quadratic constraints are hard to tackle both, from a theoretical and numerical perspective. Even if some commercial solvers are somewhat able to deal with (special cases of) such optimization problems, they internally apply suitable linearization strategies to obtain a more tractable formulation. In the relevant literature, many of these linearization techniques have been developed in the last decades, see [35] for a general overview and [34] for an application-oriented study. As common solvers are typically following one specific strategy, the numerical effects (of the chosen method) can depend on the particular optimization problem they are confronted with. Hence, in the following, we briefly present the most important linearization techniques (mainly following [34]) and directly apply them to the SBPP.

4.2.1. Glover-Wolsey-Linearization (GW)

One of the most classical ways to rephrase products of binary variables was presented in [38]. Therein, for any $(i,k) \in \Delta$ and $j \in \Delta_{ik}$, the authors introduce a new variable $\xi_{ij}^k \in \mathbb{B}$ and add the set of constraints[9]

$$\xi_{ij}^k \leq x_{ik}, \qquad \xi_{ij}^k \leq x_{jk}, \qquad \xi_{ij}^k \geq x_{ik} + x_{jk} - 1 \tag{34}$$

to the considered model. Then, it can be shown that we have $x_{ik} \cdot x_{jk} = 1$ if and only if $\xi_{ij}^k = 1$ holds, so that the new variable can replace the product at any position. By way of example, Conditions (29) are modified to

$$\sum_{(i,k)\in\Delta} \alpha_i x_{ik} - 2 \sum_{(i,k)\in\Delta} \sum_{j\in\Delta_{ik}} \mu_i \mu_j \xi_{ij}^k \leq y_k, \tag{35}$$

representing a linear inequality. Altogether, this approach requires $\mathcal{O}(n^3)$ new (binary) variables and $\mathcal{O}(n^3)$ (linear) restrictions and it was already applied to the SBPP in [59].

4.2.2. "Old" Linearization (OLD)

The probably earliest method to linearize quadratic terms was proposed by different independent research articles, see [5, 33] for publications in French language, and [81] for an English manuscript. For any $(i,k) \in \Delta$ and $j \in \Delta_{ik}$, the key idea of this approach is to also directly replace the quadratic terms $x_{ik}x_{jk}$ by a new binary variable ξ_{ij}^k and to add two inequalities

$$2\xi_{ij}^k \leq x_{ik} + x_{jk}, \qquad x_{ik} + x_{jk} \leq 1 + \xi_{ij}^k, \tag{36}$$

so that, again, $x_{ik} \cdot x_{jk} = 1$ holds if and only if $\xi_{ij}^k = 1$ is true. Moreover, it is clear that Conditions (29) will have the same structure as in (35). This approach also requires $\mathcal{O}(n^3)$ new binary variables and $\mathcal{O}(n^3)$ additional linear constraints. However, compared to the previous method by Glover and Wolsey, the total number of restrictions is smaller.

4.2.3. Glover Linearization (GLO)

The method described in [37] focuses on the coefficients $\mu_i\mu_j$ in the quadratic terms[10] appearing in (29). For any $(i,k) \in \Delta$, let us define

$$L_{ik} := \sum_{j\in\Delta_{ik}} \mu_i \mu_j$$

and an auxiliary variable $\zeta_{ik} \geq 0$. These variables are used to rephrase Condition (29) as

$$\sum_{(i,k)\in\Delta} \alpha_i x_{ik} - 2 \sum_{(i,k)\in\Delta} \zeta_{ik} \leq y_k, \tag{37}$$

meaning that ζ_{ik} has to display the information contained in the sum $\sum_{j\in\Delta_{ik}} \mu_i\mu_j x_{ik}x_{jk}$ appearing in the original quadratic conditions. This can be ensured by adding the following set of constraints

$$\sum_{j\in\Delta_{ik}} \mu_i \mu_j x_{jk} \geq \zeta_{ik}, \quad \zeta_{ik} \geq \sum_{j\in\Delta_{ik}} \mu_i \mu_j x_{jk} - L_{ik}(1 - x_{ik}), \quad L_{ik}x_{ik} \geq \zeta_{ik} \tag{38}$$

for any $(i,k) \in \Delta$, altogether leading to $\mathcal{O}(n^2)$ new (continuous) variables and $\mathcal{O}(n^2)$ linear constraints. Indeed, assuming $x_{ik} = 0$, the third condition from (38) leads to $\zeta_{ik} = 0$, so that we have $\zeta_{ik} = \sum_{j\in\Delta_{ik}} \mu_i\mu_j x_{ik}x_{jk}$ (both terms are zero) in this case. Contrariwise, for $x_{ik} = 1$, the first and second condition in (38) also leads to $\zeta_{ik} = \sum_{j\in\Delta_{ik}} \mu_i\mu_j x_{ik}x_{jk}$.

[9] Remarkably, these constraints have been shown to be facet-defining, see [64].

[10] In the original work, there are two ways to implement this linearization, either by summing over the "row indices" i or the "column indices" j. However, here the index j depends on the choice of i, so that only the presented method can be applied.

4.2.4. Oral-Kettani Linearization (OK1 & OK2)

So far, the linearizations somewhat represented "algebraically equivalent" transformations of (29) meaning that we have replaced a specific term by some new variable that exactly mimics the behavior of the original term thanks to additional constraints. Contrary to that, the following strategies apply transformations leading to a situation that could be termed as an "equivalence in the sense of optimization". More precisely, a feasible point from the original problem is possibly replaced by more than one feasible point in the linearized version, but without influencing the optimal value.

Considering again the terms L_{ik}, $(i,k) \in \Delta$, from the previous part, in [63] the following idea (termed as "OK1") is introduced. By means of auxiliary variables $\zeta_{ik} \geq 0$, $(i,k) \in \Delta$, we rephrase Condition (29) as

$$\sum_{(i,k)\in\Delta} \alpha_i x_{ik} - 2 \sum_{(i,k)\in\Delta} (L_{ik} x_{ik} - \zeta_{ik}) \leq y_k \tag{39}$$

and add constraints

$$\zeta_{ik} \geq L_{ik} x_{ik} - \sum_{j\in\Delta_{ik}} \mu_i \mu_j x_{jk} \tag{40}$$

for any $(i,k) \in \Delta$, thus requiring $\mathcal{O}(n^2)$ (continuous) variables and $\mathcal{O}(n^2)$ new restrictions (but a considerably smaller number compared to Glover's method). Assuming $x_{ik} = 0$ also $\zeta_{ik} = 0$ is a possible choice in the linearized version (leading to exactly the same contribution to the feasibility condition). As stated earlier, here the additional constraints would also allow a larger value of ζ_{ik}. Given the fact that ζ_{ik} has a positive coefficient in (39) and that we aim at minimizing the sum of the y-variables (which can be influenced by a larger value of ζ_{ik}), it becomes clear that no other feasible choice of ζ_{ik} can lead to a smaller objective value. A similar argumentation can be done for $x_{ik} = 1$. Also here, choosing the lowest possible value for ζ_{ik} (according to (40)) leads to an exact reformulation of the feasibility condition, while all other possible choices for ζ_{ik} cannot lead to strictly smaller objective values. Consequently, this linearization generates some additional feasible points, but none of those influences the optimal value.

A second version of this linearization (hereinafter referred to as "OK2"), is based on reformulating (29) as

$$\sum_{(i,k)\in\Delta} \alpha_i x_{ik} - 2 \sum_{(i,k)\in\Delta} \left(\sum_{j\in\Delta_{ik}} \mu_i \mu_j x_{jk} - \zeta_{ik} \right) \leq y_k \tag{41}$$

and adding

$$\zeta_{ik} \geq \sum_{j\in\Delta_{ik}} \mu_i \mu_j x_{jk} - L_{ik} x_{ik} \tag{42}$$

for any $(i,k) \in \Delta$, hence exhibiting the same number of additional constraints and extra variables as the first variant. Here, $x_{ik} = 1$ allows (among other possibilities) the choice $\zeta_{ik} = 0$ which exactly models the contribution of $x_{ik} = 1$ to the original feasibility condition. According to the objective function and the positive coefficients related to ζ_{ik}, any other choice cannot lead to a smaller number of active bins. In a similar way, $x_{ik} = 0$ allows ζ_{ik} to attain its lower bound thus having no contribution to the capacity constraint either (as it is prescribed by $x_{ik} = 0$). Hence, also this method describes a correct transformation of the quadratic model into a linearized one.

4.2.5. Chaovalitwongse-Pardalos-Prokopyev Linearization (CPP)

The last method to be presented here is due to [14] and closely related to Glover's strategy explained above. However, the main difference is that fewer additional restrictions are required. More precisely, we again consider the auxiliary variables $\zeta_{ik} \geq 0$, $(i,k) \in \Delta$, reformulate Conditions (29) as

$$\sum_{(i,k)\in\Delta} \alpha_i x_{ik} - 2 \sum_{(i,k)\in\Delta} \zeta_{ik} \leq y_k, \tag{43}$$

and add the constraints

$$\sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{jk} \geq \zeta_{ik}, \qquad L_{ik} x_{ik} \geq \zeta_{ik} \tag{44}$$

for any $(i,k) \in \Delta$. As clearly to be seen, the second inequality from (38) does not appear in this setting which is mainly caused by the following observation: Considering a minimization problem and having negative coefficients related to ζ_{ik}, an explicit lower bound on these variables is not required (in the case $x_{ik} = 1$) because any choice $\zeta_{ik} > \sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{jk}$ cannot lead to an improved objective value. So, dropping this condition from (38) makes the difference between an "algebraically equivalent" linearization (GLO) and the current reformulation (CPP) being "equivalent in the sense of optimization". Even if the relationship between both strategies is very close, considerably different performances have been observed in the numerical experiments reported in [34], so that we include both in our computational test.

5. Computational Experiments

In this section, we investigate the numerical properties of the various approaches mentioned in this article. Within these computational experiments, we will use the following strategy: At first, we perform extensive tests to find the (on average) best lower and upper bound for the SBPP. Then, these information will be used in the exact models, e.g., to define a reasonably small set K of possibly required bins, and to already set a favorably large amount of them to $y_k = 1$ (according to the information provided by the lower bound). Moreover, the feasible point computed by the best heuristic can be passed to Gurobi, a commercial state-of-the-art solver for mathematical optimization problems [41], to give the software a "warm start". Of course, due to the high degree of randomization within the solution steps applied by common ILP solvers, it is not clear that this will always boost the performance. However, in other fields like the *skiving stock problem* (or *dual bin packing problem*), this approach turned out to be very effective for most of the instances, see [58]. Before presenting the results of our computational tests, we first describe the instances to be considered in more details.

5.1. Data Set and Methodology

To accurately judge the performance of the presented solution approaches for different scenarios, a wide variety of instances shall be used. More precisely, the following main categories will appear in our tests:

(A) The first set is formed by randomly generated instances whose characteristics are drawn according to the instructions given in [59, Sect. 4.3]. For the sake of completeness, we state that (for given n and ε and all $i \in I$) the parameter μ_i is chosen from $[0.1, 0.5]$ based on a uniform distribution, whereas σ_i always belongs to the set

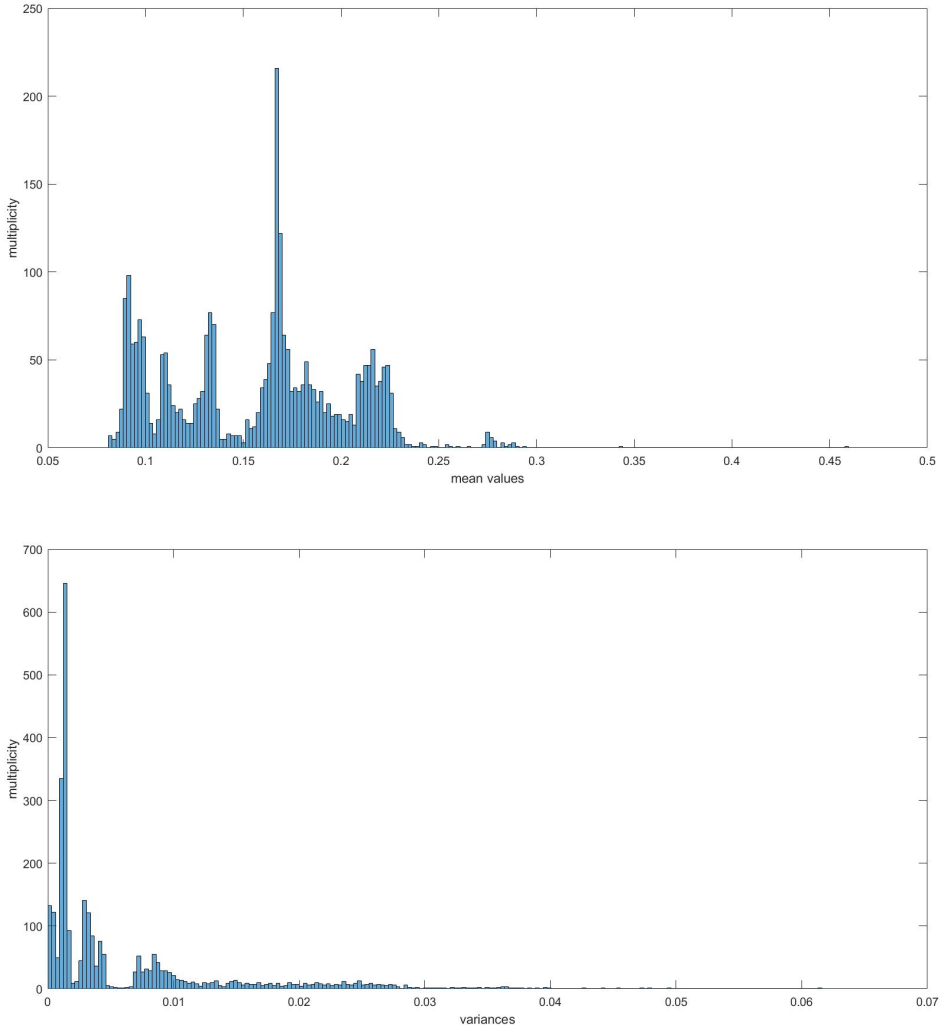
$$\left(0.01, \frac{1}{2} \cdot \min\left\{\frac{\mu_i}{q_{1-\varepsilon}}, \frac{1-\mu_i}{q_{1-\varepsilon}}\right\}\right) \overset{\mu_i \leq 0.5}{=} \left(0.01, \frac{1}{2} \cdot \frac{\mu_i}{q_{1-\varepsilon}}\right), \tag{45}$$

to (i) avoid high probabilities for negative item sizes and (ii) ensure that the SBPP is solvable (see also Remark 4).

(B) The second set contains real-world data from a Google data trace collected over a period of 30 days (in May 2011). All the details are stored in a csv-file of roughly 167 GB and can be found online, see [66] for further information. Interestingly, only approximately 0.59% of the jobs are responsible for almost 80% of the CPU utilization. Hence, here we only focus on this specific subset of 2857 jobs (and one exemplary time interval of 5 minutes) instead of dealing with the complete aforementioned file (which would be too data-intensive). Finding an optimal assignment of these representative jobs (having relatively large resource consumptions) already considerably contributes to a reduced energy

consumption. Finally, to get an overview on the characteristics of these items, we present two histograms showing the distribution of the parameters μ_i and σ_i^2, $i \in I$, see Fig. 5. As to be clearly seen, the volatility of these specific jobs is sufficiently small, so that the probability of negative item sizes is neglectable. However, when choosing $\varepsilon = 0.01$ (i.e., $q_{1-\varepsilon} \approx 2.33$) in Category (A), the average mean value $\mu_A = 0.3$ leads to an average variance $\sigma_A^2 \approx 0.004$, so that the instances from Category (B) can possess larger variances than those of Category (A). Moreover, the vast majority of the mean values in Category (B) is located in the interval $[0.1, 0.3]$, thus forming instances with typically smaller item sizes (compared to Category (A)) and much more feasible combinations. Thus the instances of Category (B) are considerably different from those of Category (A).

Figure 5: Distribution of the mean values μ_i (upper histogram) and variances σ_i^2 (lower histogram) for the considered set of 2857 jobs



Remark 20. *Note that there are no benchmark instances specifically designed for the SBPP. Theoretically, we could have equipped the classical sets for the BPP, which have been collected in the BPPLIB [29] and can be found on the Internet by* `http://or.dei.unibo.it/library/bpplib`*, with an additional randomly drawn parameter σ_i, $i \in I$, but finally we decided against it. On the one hand, these parameters require an interval similar to that of Category (A) to keep*

the probability of negative item sizes very low without compromising too much on the volatility, so that the modified benchmarks would not cover completely new scenarios within the computations. On the other hand, we also aim at adressing the exact solution of the instances considered here which would be impossible given the instance sizes of most of these benchmarks.

5.2. Results for the Lower Bounds

Throughout this subsection, we consider the values

$$\begin{aligned} n &\in \{10, 15, 20, 50, 100, 200, 300, 500\}, \\ \varepsilon &\in \{0.01, 0.05, 0.1, 0.2\}. \end{aligned}$$

To facilitate comparing the results also for different error bounds ε, note that (for fixed n) we always generated one set of 20 instances which is then used for any of the four possible choices of ε. Consequently, for Category (A), we had to draw the values σ_i, $i \in I$, from the most restrictive interval in (45), i.e., for $\varepsilon = 0.01$. In the following tables (Tab. 2 and 3), we display the average results of the three lower bounds presented in Subsect. 3.1, i.e., the bound lb_1 from [59], and the two new bounds lb_2 and lb_3. For any pair (n, ε) of input parameters, we use boldface to highlight the best performing approach. Here, by *best* we mean the bound achieving the largest (average) value.

The following main observations can be made:

- Obviously, the values of the lower bounds grow for increasing n and decreasing ε. It is clear that either more items or a more restrictive assignment policy (caused by a lower error bound) lead to more active servers.
- The bound lb_3 performs best for any considered set of instances. Moreover, both relations between the values lb_1 and lb_2 can be witnessed. More precisely, for relatively small numbers of items n, the bound lb_1 is better than lb_2, whereas for larger values of n the opposite is true. Note that this general behavior has already been discussed in the theoretical part of the paper (see Subsect. 3.1), so that the numerical results second the observations made before.
- Having a look at a fixed parameter setting (n, ε) and a fixed lower bound, then the corresponding values in Category (A) are always larger than those of Category (B). As mentioned earlier, the input data μ_i, $i \in I$, in Category (B) are typically smaller than in Category (A), so that also the bounds attain smaller values.
- Interestingly, especially for larger instances, a remarkable difference is noticeable in the relative gap between the two best lower bounds (lb_2 and lb_3). By way of example, let us consider the data for $(n, \varepsilon) = (500, 0.01)$. Here, for the real-world jobs we see a relative gap $(lb_3 - lb_2)/lb_2$ of roughly 21%, while the same formula results to only 12% for the randomly generated instances. A similar (though not as clear-cut) trend can also be observed when comparing other instance sets from Tab. 2 and Tab. 3, the reason of which we attribute to the tightness of Inequality (14) that was used to construct lb_2. For the rather small mean values in Category (B), (14) represents a very rough estimation, while for instances of Category (A), the performance is better (on average) due to the larger values μ_i, $i \in I$.

To better illustrate the above observations and the general behavior of the three lower bounds, for two representative error bounds we also added a graphical visualization of the data contained in Tab. 2 and 3 in Fig. 6. Overall, as it should be expected, we will choose $\eta = lb_3$ as the best lower bound for the preprocessing in the exact approaches.

ε	lb_1	lb_2	lb_3	lb_1	lb_2	lb_3
$n = 10$				$n = 100$		
0.01	3.60	3.50	**3.85**	28.85	29.45	**32.90**
0.05	3.50	3.30	**3.60**	28.55	28.70	**31.35**
0.10	3.50	3.30	**3.60**	28.50	28.35	**30.55**
0.20	3.40	3.20	**3.50**	28.30	28.15	**29.65**
$n = 15$				$n = 200$		
0.01	4.85	4.65	**5.25**	56.40	58.05	**64.95**
0.05	4.75	4.40	**5.05**	55.95	56.50	**61.90**
0.10	4.70	4.40	**4.90**	55.75	55.95	**60.20**
0.20	4.45	4.40	**4.70**	55.55	55.45	**58.40**
$n = 20$				$n = 300$		
0.01	6.70	6.50	**7.40**	84.95	87.60	**98.40**
0.05	6.45	6.45	**6.90**	84.35	85.45	**93.60**
0.10	6.45	6.40	**6.80**	84.15	84.55	**91.30**
0.20	6.45	6.35	**6.60**	83.95	83.85	**88.45**
$n = 50$				$n = 500$		
0.01	15.45	15.45	**17.35**	140.00	145.05	**162.60**
0.05	15.30	15.25	**16.45**	139.40	141.50	**154.95**
0.10	15.30	15.15	**16.30**	139.05	140.05	**150.90**
0.20	15.15	14.85	**15.55**	138.65	138.80	**146.30**

Table 2: Numerical results for the lower bounds corresponding to Category (A)

ε	lb_1	lb_2	lb_3	lb_1	lb_2	lb_3
$n = 10$				$n = 100$		
0.01	2.70	2.40	**2.90**	17.80	19.90	**23.95**
0.05	2.35	2.15	**2.70**	17.25	18.00	**21.15**
0.10	2.30	2.00	**2.35**	17.10	17.30	**19.90**
0.20	2.05	2.00	**2.25**	16.70	16.60	**18.40**
$n = 15$				$n = 200$		
0.01	3.55	3.40	**3.95**	34.70	40.00	**48.40**
0.05	3.20	3.00	**3.60**	33.85	36.00	**42.60**
0.10	3.00	3.00	**3.40**	33.45	34.45	**40.00**
0.20	**3.00**	**3.00**	**3.00**	33.00	33.05	**36.95**
$n = 20$				$n = 300$		
0.01	4.30	4.30	**5.20**	50.55	59.40	**71.55**
0.05	4.00	4.00	**4.70**	49.75	53.50	**63.30**
0.10	4.00	4.00	**4.30**	49.20	51.05	**59.25**
0.20	**4.00**	**4.00**	**4.00**	48.55	49.00	**54.75**
$n = 50$				$n = 500$		
0.01	9.90	10.40	**12.55**	83.15	98.90	**119.65**
0.05	9.35	9.40	**11.15**	81.95	89.05	**105.65**
0.10	9.20	9.05	**10.45**	81.40	85.15	**98.90**
0.20	8.95	8.85	**9.90**	80.55	81.60	**91.50**

Table 3: Numerical results for the lower bounds corresponding to Category (B)

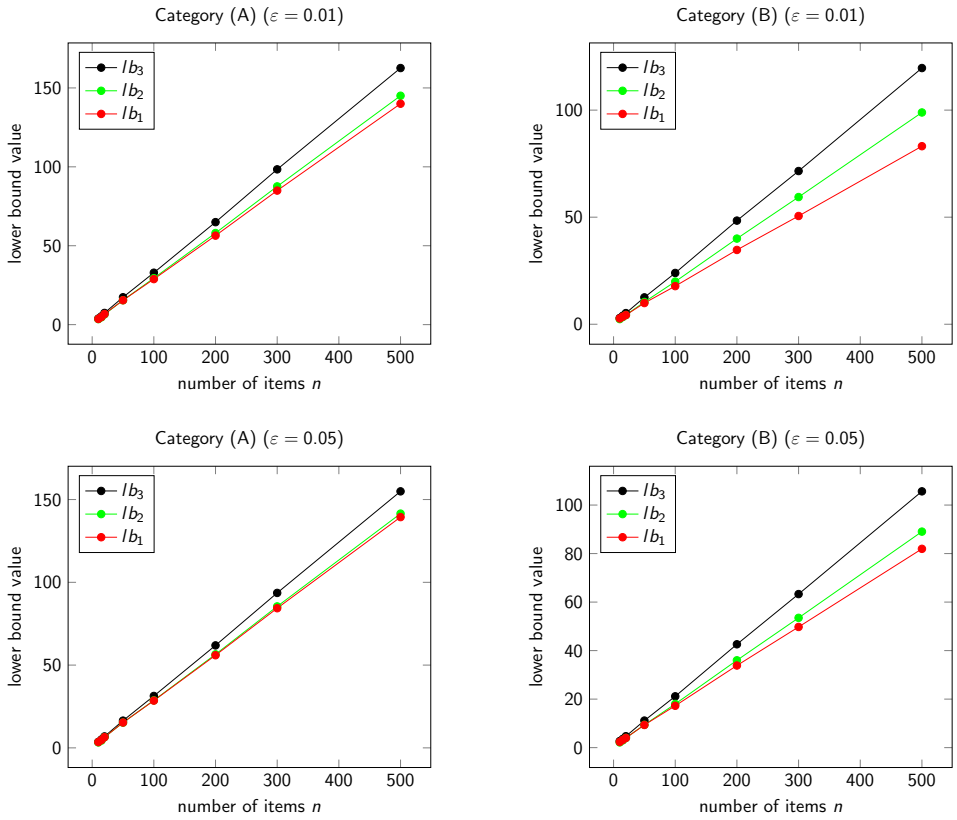


Figure 6: Comparative illustration of the three lower bounds for two representative choices of ε

5.3. Results for the Upper Bounds

We consider the same parameter constellations (for n and ε) as in the previous subsection and include the following heuristic approaches into our comparison:

- FFD heuristic from [59] leading to ub_1,
- FFD with alternative sorting, see Alg. 4, leading to ub_2,
- Fractional NFD with FFD post processing, see Alg. 3, leading to ub_3,
- Local Search method from [21] leading to ub_4.

Note that we only focus on the most promising approaches and, hence, do not cope with the rather naive approximate solution strategies reported in the early publications [15, 80]. In Tab. 4 and 5, the average values of the various upper bounds have been collected, the best of which is again printed in boldface. In this setting, *best* refers to the minimum average value.

The main observations can be summarized as follows:

- There is almost no need to say that the upper bound values are growing here as well, whenever n is increased or ε is decreased.
- In all the simulations, at least ub_1, ub_2, and ub_4 are quite close together, so that (from a practical point of view) they are roughly equivalent. However, we highlight the fact that the state-of-the-art algorithm from the recent literature (i.e., the local search method from [21]), only performed best for one single set of 20 very small instances in Tab. 5.

ε	ub_1	ub_2	ub_3	ub_4	ub_1	ub_2	ub_3	ub_4
$n = 10$					$n = 100$			
0.01	**4.05**	4.10	4.30	4.15	**33.55**	34.15	37.00	34.70
0.05	**3.70**	3.80	4.15	3.85	**31.95**	32.55	34.60	33.30
0.10	**3.65**	3.70	4.10	3.65	**31.00**	31.70	33.65	32.00
0.20	**3.60**	3.70	3.75	3.65	**30.05**	30.80	32.50	30.90
$n = 15$					$n = 200$			
0.01	**5.35**	5.50	5.90	5.55	**66.05**	67.35	72.75	67.95
0.05	**5.20**	**5.20**	5.55	5.35	**62.90**	63.95	67.40	64.50
0.10	**5.00**	5.10	5.45	5.30	**61.10**	62.15	65.80	62.70
0.20	**4.85**	5.00	5.20	5.00	**59.00**	60.20	63.55	60.45
$n = 20$					$n = 300$			
0.01	**7.60**	7.70	8.60	7.95	**100.00**	101.70	107.30	102.65
0.05	**7.25**	7.40	8.15	7.50	**95.00**	96.70	101.20	97.45
0.10	**7.00**	7.20	7.70	7.30	**92.30**	94.20	98.50	94.60
0.20	**6.80**	6.95	7.40	7.10	**89.30**	91.30	95.30	91.50
$n = 50$					$n = 500$			
0.01	**17.90**	18.20	19.85	18.50	**165.30**	167.75	176.45	168.80
0.05	**16.95**	17.25	18.75	17.70	**157.00**	159.60	165.55	160.35
0.10	**16.40**	16.90	18.10	17.20	**152.55**	155.50	161.95	156.05
0.20	**16.00**	16.25	17.35	16.80	**147.50**	150.65	157.60	150.85

Table 4: Numerical results for the upper bounds corresponding to Category (A)

ε	ub_1	ub_2	ub_3	ub_4	ub_1	ub_2	ub_3	ub_4
$n = 10$					$n = 100$			
0.01	3.20	**3.00**	3.15	3.05	25.05	**24.85**	25.80	25.60
0.05	**2.80**	**2.80**	**2.80**	2.85	**22.15**	22.25	22.85	22.45
0.10	**2.45**	2.50	2.65	2.60	**20.60**	**20.60**	20.95	21.10
0.20	2.35	**2.30**	2.40	**2.30**	**19.15**	19.25	19.60	19.30
$n = 15$					$n = 200$			
0.01	**4.20**	4.30	4.35	4.25	**50.50**	**50.50**	51.85	51.65
0.05	**3.75**	**3.75**	3.80	3.80	**44.60**	44.65	45.60	45.50
0.10	3.55	**3.45**	3.60	3.60	**41.50**	41.55	42.25	42.35
0.20	**3.10**	3.15	3.15	3.25	**38.40**	**38.40**	39.10	39.00
$n = 20$					$n = 300$			
0.01	**5.50**	**5.50**	5.75	5.55	74.75	**74.50**	76.75	76.55
0.05	**4.90**	**4.90**	5.00	4.95	**65.85**	66.15	67.55	67.30
0.10	**4.55**	4.60	4.80	4.65	**61.40**	**61.40**	62.60	62.80
0.20	**4.00**	4.05	4.20	4.10	56.85	**56.75**	58.00	57.80
$n = 50$					$n = 500$			
0.01	13.30	**13.25**	13.65	13.55	125.30	**124.45**	128.30	127.85
0.05	**11.80**	**11.80**	12.10	12.00	**110.05**	110.45	112.75	112.30
0.10	**11.00**	**11.00**	11.25	11.15	**102.25**	102.60	104.30	104.75
0.20	**10.05**	10.10	10.25	10.15	**94.70**	**94.70**	96.45	96.15

Table 5: Numerical results for the upper bounds corresponding to Category (B)

In all other scenarios, either the ordinary FFD algorithm from [59] or the FFD approach based on an alternative item order, see Alg. 4, won. More precisely, while for Category (A) the ordinary FFD heuristic seems to be the best, for the instances from Category (B) we clearly notice advantages for both techniques mentioned before. We attribute this observation to the fact, that the typically smaller mean values of the real-world instances allow for much more feasible combinations with very good capacity utilization. Hence, for these instances, different ways of presorting can lead to comparably good results as it is supported by the computations.

- In addition to the previous point, we underline that the local search method from [21] almost always performed like an ordinary first fit approach. Only for two single instances from Category (B), the subsequent post-processing steps performed by this algorithm led to any benefits (so that one bin could be saved).

- Especially for Category (A), but also for many cases in Category (B), the method proposed by Alg. 3 did not lead to convincing results even though it has the best theoretical performance guarantee. Remember that in this strategy, we first applied the fractional next fit decreasing heuristic and then first excluded the fractional items from these bins, while in a second step, we wanted the recombined complete items to fill these gaps (in an FFD style). However, particularly for Category (A) dealing with larger item sizes, it turned out that the empty space resulting from the removal of fractional items cannot be filled again later. Consequently, this algorithm typically created worse bin utilizations than the other heuristics ending up with the poorest upper bound in most cases.

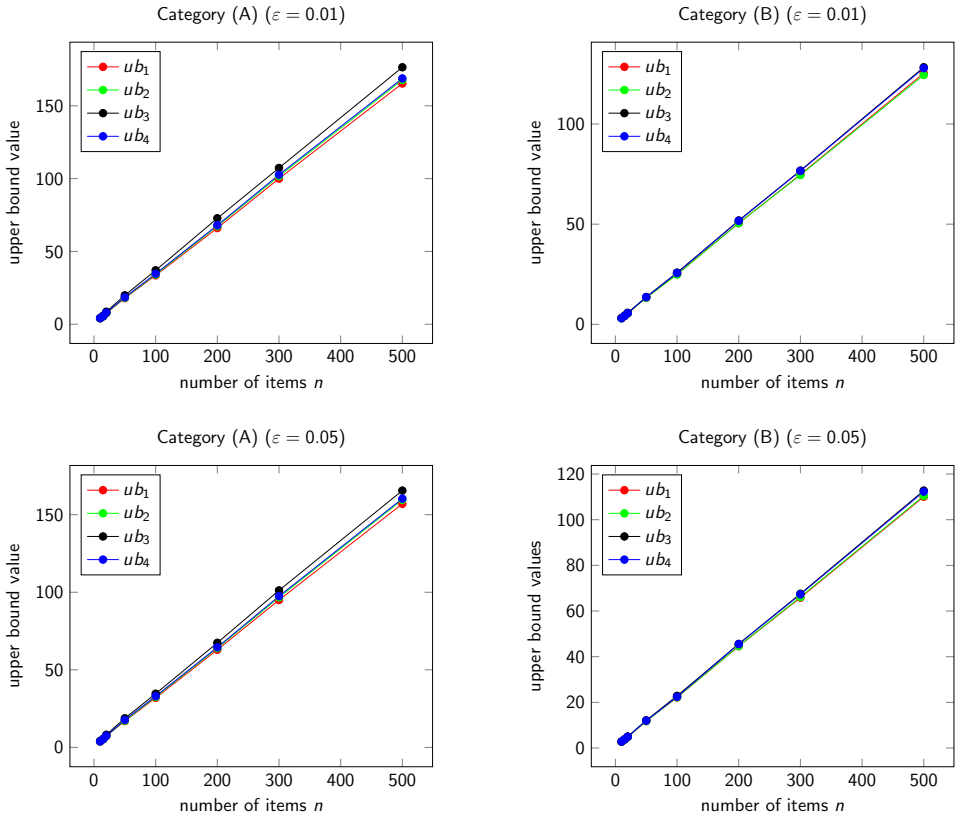


Figure 7: Comparative illustration of the four upper bounds for two representative choices of ε

As for the study dealing with the lower bounds, also here we refer the interested reader to a

graphical illustration clearly showing the trends described in the above list, see Fig. 7. Overall, given the technicalities (leading to the complexity $\mathcal{O}(n^3)$) and the numerical performance of the local search method from [21], for arbitrary instances we clearly recommend to use one of (or both) the heuristics described in [59] and Alg. 4 to obtain a fast and good feasible solution.

As a brief summary of the impressions obtained so far, we would like to point out the very small difference between the smallest upper bound and the largest lower bound. In many cases, these two values already form a rather narrow interval for the currently still unknown optimal value of an instance. To better visualize this observation, Fig. 8 shows the (average) relative optimality gap $\Delta_r \in [0, 1]$, defined by

$$\Delta_r(E) := \frac{\min\{ub(E)\} - \max\{lb(E)\}}{\min\{ub(E)\}},$$

for the instances E from both categories. Note that, here, we first calculated the value Δ_r for every single instance and then built the averages (meaning that we did not simply use the averages contained in the previous tables for a whole set of 20 instances).

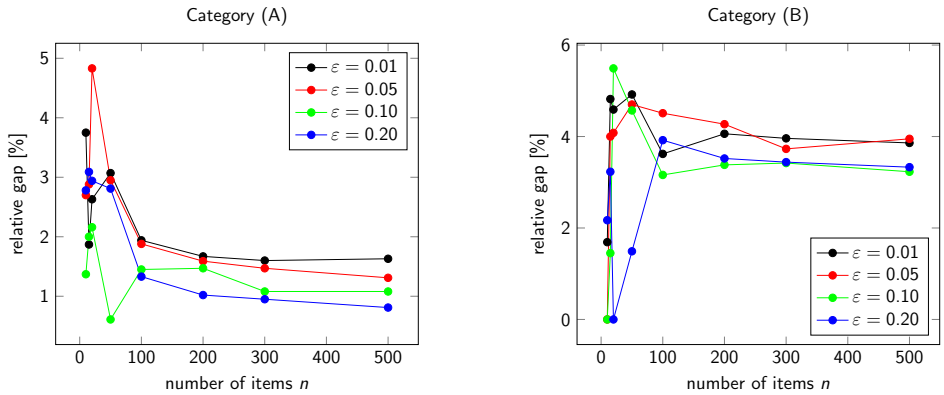


Figure 8: Relative optimality gaps for the instances of Category (A) and (B)

The following interesting facts can be observed in Fig. 8:

- While for smaller values of n, some "oscillations" of the graphs can be witnessed, in the long-term evolution a relatively constant behavior is manifested. Having in mind that already one unit distance between $\min ub$ and $\max lb$ can lead to substantial relative gaps Δ_r (for small values of n), the shape and the general trend of the curves are plausible.

- Remarkably, the relative gaps for the more challenging larger instances turn out to be very small resulting in roughly $1 - 2\%$ for Category (A) and approximately $3 - 4\%$ for Category (B). Again, remember that the absolute values (of the bounds) for Category (B) are considerably smaller than those of Category (A), so that the slight difference in terms of the relative gaps can be justified. Overall, these relative gaps clearly demonstrate the very good performance of both, the lower and the upper bounds, and additionally state that the feasible points provided by the best performing heuristic is already very close to an optimal solution (with respect to the objective value).

- Although (at least partly) suggested by the left picture in Fig. 8, there is no general correlation between ε and the relative gap. By way of example, we illustrate this by one instance from Category (A) having $n = 20$ items, see Tab. 6. Therein, it is clearly shown that an increasing value of ε can lead to both, an increase or a decrease in terms of Δ_r.

	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.10$	$\varepsilon = 0.20$
$\min ub$	8	8	8	7
$\max lb$	8	7	7	7
Δ_r	0	0.125	0.125	0

Table 6: Development of the best lower and upper bound (depending on ε) for one specific instance

5.4. Results for the Exact Approaches

When addressing the exact solution of the problem under consideration, the results from the previous investigations can be very helpful. While a lower bound can be used to already fix some y-variables, knowing a heuristic solution determines the cardinality of the set K (i.e., the maximum number of servers) and, in addition, offers the possibility of passing an initial feasible point to the solver. Note that the latter particularly requires to

(i) renumber the bins in the heuristic solution so that the only index pairs $(i,k) \in \Delta$ occur: This can be done by a fairly simple method searching for the bin containing item $i = 1$ and changing its index to $k = 1$. Afterwards, the following items are processed in the same way meaning that, whenever the current item i appears in a bin that is not yet labeled, then this bin receives the lowest possible bin number that is still available. Otherwise, if the current item belongs to a bin that already got its new label before, then we just move forward to the next item.

(ii) also fix the auxiliary variables of the respective linearization: As a result of our theoretical explanations in Sect. 4, this can be done by setting

$$\xi_{ij}^k := \begin{cases} 1, & \text{if } x_{ik} = 1 \text{ and } x_{jk} = 1, \\ 0, & \text{otherwise,} \end{cases}$$

for (GW) and (OLD),

$$\zeta_{ik} := \begin{cases} \sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{jk}, & \text{if } x_{ik} = 1, \\ 0, & \text{otherwise,} \end{cases}$$

for (GLO) and (CPP),

$$\zeta_{ik} := \begin{cases} L_{ik} - \sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{jk}, & \text{if } x_{ik} = 1, \\ 0, & \text{otherwise,} \end{cases}$$

for (OK1), or

$$\zeta_{ik} := \begin{cases} \sum_{j \in \Delta_{ik}} \mu_i \mu_j x_{jk}, & \text{if } x_{ik} = 1, \\ 0, & \text{otherwise,} \end{cases}$$

for (OK2).

In this subsection, we compare the linearizations presented in Sect. 4, any of which being operated in two modes: a *cold start* (i.e., no feasible point is initially defined) and a *warm start* (i.e., the best feasible point resulting from the four heuristics investigated before is given to the solver).

Remark 21. *Here we do not intend to add the valid inequalities $x_{ik} \leq y_k$, $(i,k) \in \Delta$, or $y_k \geq y_{k+1}$, $k \in \{1, \ldots, |K| - 1\}$, to the ILP models, since in our internal precalculations for some representative test sets, none of these restrictions led to any (substantial) gains in terms of the solution times. Although this may seem surprising at first glance, there are quite simple explanations for this observation:*

- *Originally, the inequalities of type $x_{ik} \leq y_k$ were introduced to strengthen the LP relaxation of any exact approach. While such effects may actually occur in an unimproved model, fixing the y-variables according to the lower bound η is much more important for the LP value. More precisely, in our computations the LP value did never change after having added the valid inequalities, so that only the model's complexity is increased by a quadratic number of restrictions.*
- *In a similar way, the vast majority of the inequalities of type $y_k \geq y_{k+1}$ (aiming to reduce the number of symmetric solutions) is already satisfied by fixing the first η variables to one. Besides, given the very narrow interval formed by the best lower and upper bounds, only a very few constraints of this type would remain, the latter of which never showed any considerable positive effects in our precalculations.*

Because of the large number of different setups (i.e., four different choices of ε, six different linearizations, cold start vs. warm start), we mainly focus on the solution times and the number of instances solved to optimality within a given time limit. Since in practice decisions have to be made rather quickly, we start with $t_{\max} = 300$ seconds. Later in Remark 22, when also reporting about some larger instance sizes, we will briefly discuss the effects of an enlarged time limit (600 seconds). Note that, depending on the particular application scenario, real-world time limits can be much stricter (e.g., when jobs arrive in an online fashion), so that in these cases only heuristic solutions are of interest, in general. Addressing the exact solution of the problem under consideration obviously requires a slightly more relaxed time limit, but this does not represent an unrealistic assumption. By way of example, the Google data trace [66] (forming the instances of Category (B)), repeatedly specifies the parameters μ_i and σ_i for intervals of 5 minutes length, so that this can be seen as the maximum tolerable time to make the decision based on the given information.

To present at least some data for the number of constraints and variables, we only refer to Fig. 9 showing the development of these indicators for moderate instances sizes.

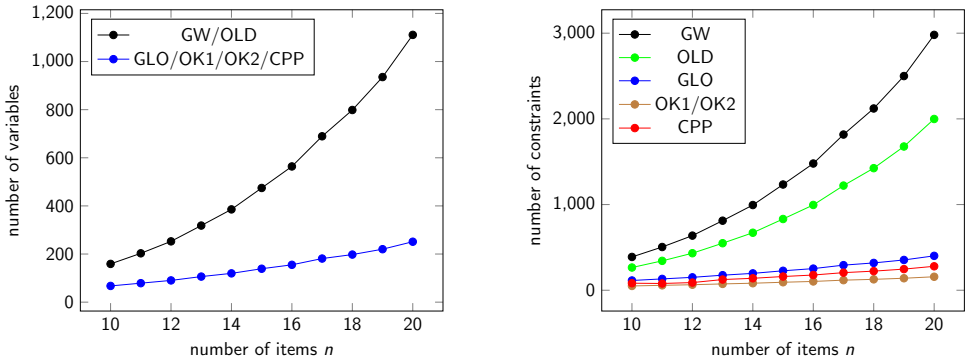


Figure 9: A trend for the numbers of variables and constraints in all linearization techniques, here only exemplified for Category (A) and $\varepsilon = 0.05$

These numerical data confirm the statement (already made in the theoretical investigations in Sect. 4) that the various linearization techniques can be roughly divided into two groups.

(1) The methods (GW) and (OLD) possess a cubic growth in the numbers of variables and constraints, with the latter having a constantly lower value for (OLD) because this strategy just requires two (instead of three) additional inequalities per auxiliary variable.

(2) The methods (GLO), (OK1), (OK2), and (CPP) only require a quadratic number of variables and constraints. Furthermore, we clearly notice that (OK1) and (OK2) create one

additional constraint per artificial variable, while (CPP) has a "factor" of two and (GLO) a "factor" of three.

Due to these characteristics, the representatives of the second groups are expected to show better computational results (on average), e.g., in terms of the solution times. To this end, we now investigate the numerical performance of the exact approaches in more details. Since we are basically dealing with a Kantorovich-type model, finding an exact solution of the SBPP is likely to be successful only for moderately sized instances. As already indicated by the numbers of items in Fig. 9, here we mainly consider constellations in the lower double-digit range, not without pointing out that in the reference article [59], already instances with $n = 18$ items required more than 10 minutes on average for the fairly large (thus, computationally rather favorable) error bound $\varepsilon = 0.25$. In the following tables, the average computation times for instances of Category (A), see Tab. 7, and Category (B), see Tab. 8, are listed for six different linearizations and both options, with (✓) and without (✗) initialization by passing the feasible point obtained by the best heuristic to the solver. Whenever there is a bracketed number behind the solution time, then this indicates the number of instances solved to optimality (from 20 instances in each test). We highlight the fact, that when reaching the time limit $t_{\max} = 300$ seconds, then this time will be counted (as a lower bound for the true computation time) to build the averages. Any model was coded in C++ and solved by Gurobi (version 9) with default settings on a computer having 8 GB RAM and a dual-core Intel Core i7-4510U with 2.6 GHz.

At first glance, the numerical results reported in Tab. 7 and Tab. 8 seem to be very surprising and a bit "chaotic". To this end, before going into a more detailed assessment of these data, let us briefly discuss what we observed as the main challenges when solving an instance with the Kantorovich-type models we use:

- The most crucial point is already determined in the preprocessing part, meaning that whenever the interval formed by the best lower bound and the best upper bound is small (meaning that there is either no difference or the difference is only one unit), then the chances of computing a solution rapidly are actually quite good. However, the fact whether such a favorable situation appears is always instance-specific. By way of example, increasing the parameter ε can also lead to a more challenging instance because the interval given by the two bounds can be enlarged, as already reported in Tab. 6.

- Even if the two bounds possess a very small (but positive) distance, solving the instance can still be very cumbersome. The main reason for this is the poor quality of the LP relaxation mostly simply mimicing the lower bound η at many nodes of the branch-and-bound tree. With this in mind, also passing a feasible point to the solver does not necessarily have to be helpful since we observed that, for these type of assignment models, finding integer points is much less problematic than rising the bounds.

- In general, a larger number of variables and constraints is connected to an increase of the numerical efforts required for the solution of an instance. However, any of the two previous points can easily influence this observation and also turn it into the opposite effect.

All these points contribute to the fact that we cannot always see a clear trend in the tables corresponding to the exact approaches. Roughly speaking, in many cases there seem to be two major classes of instances: those that are solved almost immediately in (or close to) the root node, and those that really enter the branch-and-bound algorithm (mostly without being finished within the time limit). Moreover, we would like to further stress that at least two random factors are also relevant within the solution process. On the one hand, the Gurobi solver is based on some randomization (e.g., in terms of which internal steps are performed and in which sequence); on the other hand, a randomly drawn set of 20 instances can always contain an arbitrary number of challenging instances.

Apart from these general facts to justify the possibly unexpected behavior of the numerical data, the main observations related to Tab. 7 and 8 are the following:

ε	Init.	(GW)	(OLD)	(GLO)	(OK1)	(OK2)	(CPP)
$n = 10$							
0.01	✗	0.052	0.041	0.024	0.022	0.021	0.017
	✓	0.040	0.028	0.016	0.017	0.016	0.013
0.05	✗	0.029	0.030	0.012	0.014	0.012	0.012
	✓	0.017	0.019	0.007	0.009	0.010	0.009
0.10	✗	0.023	0.020	0.011	0.011	0.010	0.010
	✓	0.014	0.015	0.007	0.008	0.008	0.010
0.20	✗	0.022	0.009	0.009	0.008	0.006	0.008
	✓	0.007	0.007	0.007	0.005	0.005	0.007
$n = 12$							
0.01	✗	0.050	0.038	0.033	0.030	0.030	0.022
	✓	0.013	0.008	0.012	0.008	0.014	0.008
0.05	✗	0.027	0.043	0.011	0.014	0.009	0.013
	✓	0.007	0.023	0.005	0.004	0.007	0.009
0.10	✗	0.117	0.096	0.030	0.031	0.034	0.035
	✓	0.105	0.083	0.036	0.028	0.025	0.024
0.20	✗	0.077	0.059	0.027	0.020	0.028	0.026
	✓	0.070	0.066	0.016	0.020	0.018	0.016
$n = 14$							
0.01	✗	1.865	2.040	0.501	0.667	0.725	0.438
	✓	2.295	1.807	0.441	0.607	0.781	0.421
0.05	✗	1.000	0.984	0.214	0.284	0.303	0.199
	✓	1.010	0.909	0.181	0.248	0.280	0.186
0.10	✗	1.049	0.987	0.240	0.187	0.230	0.168
	✓	0.937	0.963	0.204	0.195	0.185	0.138
0.20	✗	0.679	0.037	0.013	0.018	0.026	0.013
	✓	0.003	0.004	0.003	0.004	0.004	0.004
$n = 16$							
0.01	✗	6.210	3.374	1.142	2.085	2.263	1.071
	✓	4.164	3.021	1.330	2.025	2.661	**1.007**
0.05	✗	0.679	0.676	0.189	0.224	0.264	0.149
	✓	0.643	0.547	0.142	0.210	0.228	**0.101**
0.10	✗	5.212	6.394	1.199	1.655	1.781	1.176
	✓	5.257	5.130	1.185	1.637	1.920	**1.060**
0.20	✗	1.749	1.756	**0.292**	0.397	0.432	0.340
	✓	2.375	1.857	0.380	0.405	0.405	0.330
$n = 18$							
0.01	✗	17.056 (19)	16.744 (19)	12.338	14.160	15.496	**7.935**
	✓	17.046 (19)	18.058 (19)	9.013	14.683	11.251	10.045
0.05	✗	31.746 (18)	30.948 (18)	8.807	11.274	17.801	8.767
	✓	30.993 (18)	32.228 (18)	8.914	12.192	17.311	**8.407**
0.10	✗	0.247	0.299	0.065	0.064	0.045	0.068
	✓	0.217	0.388	0.022	**0.016**	0.064	0.021
0.20	✗	16.484 (19)	16.761 (19)	1.719	1.780	5.651	**1.028**
	✓	16.840 (19)	16.411 (19)	2.419	2.330	4.079	1.303

Table 7: Numerical results for the exact approaches applied to the instances of Category (A). Given the very small solution times, boldface is only used for $n \geq 16$ to indicate the best formulation.

ε	Init.	(GW)	(OLD)	(GLO)	(OK1)	(OK2)	(CPP)
$n = 10$							
0.01	✗	0.014	0.029	0.008	0.018	0.008	0.013
	✓	0.007	0.006	0.004	0.009	0.006	0.006
0.05	✗	0.009	0.009	0.007	0.010	0.009	0.008
	✓	0.003	0.004	0.003	0.004	0.003	0.004
0.10	✗	0.008	0.012	0.006	0.006	0.005	0.008
	✓	0.004	0.004	0.004	0.003	0.005	0.005
0.20	✗	0.017	0.016	0.009	0.011	0.011	0.008
	✓	0.013	0.013	0.008	0.011	0.009	0.008
$n = 12$							
0.01	✗	0.380	0.293	0.119	0.091	0.107	0.110
	✓	0.360	0.322	0.093	0.090	0.102	0.104
0.05	✗	0.012	0.019	0.008	0.011	0.007	0.009
	✓	0.003	0.003	0.004	0.004	0.003	0.005
0.10	✗	0.012	0.014	0.006	0.007	0.005	0.005
	✓	0.003	0.004	0.004	0.004	0.004	0.004
0.20	✗	0.019	0.018	0.011	0.012	0.008	0.013
	✓	0.019	0.015	0.009	0.010	0.010	0.012
$n = 14$							
0.01	✗	0.162	0.593	0.061	0.037	0.040	0.107
	✓	0.118	0.339	0.026	0.025	0.033	0.020
0.05	✗	0.080	0.082	0.039	0.056	0.028	0.036
	✓	0.074	0.052	0.025	0.023	0.030	0.024
0.10	✗	0.020	0.029	0.022	0.014	0.021	0.014
	✓	0.005	0.012	0.008	0.005	0.004	0.009
0.20	✗	0.013	0.012	0.010	0.008	0.013	0.006
	✓	0.003	0.003	0.004	0.003	0.003	0.005
$n = 16$							
0.01	✗	44.462 (19)	35.510 (18)	8.964	**6.993**	8.620	8.386
	✓	31.097 (19)	32.903 (19)	7.740	7.500	8.713	8.470
0.05	✗	0.230	0.224	0.085	0.120	0.085	0.110
	✓	0.300	0.193	0.062	0.044	0.072	**0.041**
0.10	✗	0.042	0.053	0.029	0.022	0.026	0.012
	✓	0.004	**0.003**	0.004	0.004	0.004	0.004
0.20	✗	0.059	0.073	0.020	0.030	0.049	0.033
	✓	0.035	0.060	0.020	0.017	0.010	**0.007**
$n = 18$							
0.01	✗	30.797 (18)	31.298 (18)	18.026 (19)	**15.467 (19)**	16.276 (19)	15.707 (19)
	✓	30.762 (18)	20.867 (19)	18.325 (19)	17.293 (19)	17.281 (19)	17.501 (19)
0.05	✗	15.154 (19)	15.160 (19)	4.133	6.592	15.063 (19)	1.888
	✓	15.005 (19)	15.005 (19)	**0.380**	5.007	1.878	0.664
0.10	✗	15.178 (19)	15.135 (19)	15.087 (19)	15.053 (19)	15.066 (19)	15.040 (19)
	✓	15.024 (19)	15.074 (19)	**15.008 (19)**	**15.008 (19)**	15.018 (19)	15.010 (19)
0.20	✗	0.045	0.036	0.023	0.024	0.023	0.019
	✓	0.025	0.009	**0.004**	0.005	0.008	0.012

Table 8: Numerical results for the exact approaches applied to the instances of Category (B). Given the very small solution times, boldface is only used for $n \geq 16$ to indicate the best formulation.

- In both categories, any instance with up to $n = 14$ items could be solved in much less than the given time limit. For Category (A), this also holds for instances with $n = 16$, while in Category (B) the models having a cubic number of variables and constraints already start to struggle with a few instances. Besides the reasons previously mentioned, we again underline that the mean values in Category (B) are much smaller, thus leading to much more feasible item combinations and optimization problems typically harder to solve.
- In any scenario, the second group of linearizations (i.e., those techniques only requiring a quadratic number of additional constraints and variables) is able to solve at least the same number of instances (partly even more) as the first group. Moreover, even if both groups manage to tackle the same number of instances successfully, the solution times within the second group are usually better, see the case $n = 16$ in Tab. 7.
- We did not find a single instance, that could be solved without initialization, but became impossible to solve (within the given time limit) after having added a feasible starting point. However, for two instances and formulation (OLD) in Tab. 8, the opposite effect could be noticed, meaning that passing an initial starting point to the solver made the instances easier to handle.
- By and large, (CPP) can be considered as the most appropriate formulation for almost all instance sets from Category (A), while in Category (B) also (GLO) and (OK1) performed best (on average) for some sets of instances. Actually, these were the three approaches possessing the fewest variables and constraints (see Fig. 9), so the numerical behavior observed here is based on their theoretical properties.

Remark 22. *To discuss some further aspects of the exact solution approaches, in Tab. 9 we summarize the average results (with respect to* 20 *instances each) for* $n = 20$ *items. Note that, to account for the larger instance sizes, here we selected a time limit of* $t_{\max} = 600$ *seconds. However, as clearly to be seen, extending the allowed solution time does not lead to a remarkable increase in the number of instances solved to optimality, in general. Having in mind that an unsolved instance contributes with* $600/20 = 30$ *seconds to any average value, we conclude that in many cases, finding the exact solution of an instance is either done within roughly the first* 30 *seconds or impossible within the given time limit*[11]*. In fact, only two instances benefited from doubling the time limit, meaning that we detected a solution time between* 5 *and* 10 *minutes.*
Furthermore, the (CPP) linearization seems to be the most suitable on average, since it almost always belongs to those methods that solve most instances to optimality, and, in many cases, it also offers the best (or one of the best) computing times. Only for $\varepsilon = 0.01$*, (CPP) is considerably worse than (OK1) or (OK2), respectively. Due to the very strict error bound, these instances typically require the largest number of servers to be considered in the integer model (leading to both, more variables and constraints compared to other choices of* ε*). Hence, the fact that for (OK1) and (OK2), the model complexity does not rise so sharply as for other formulations, see Fig. 9, can be one possible reason for their convincing numerical results.*

As a last point, we mention that, except for (OK1), any column contains at least one instance where adding a feasible starting point turned out to have negative effects (in such a way, that after having done the initialization, the instance could not be solved to optimality anymore). We attribute this phenomenon to the high degree of randomization in the internal Gurobi solution steps.

Our final experiment is, so to speak, in the tradition of the computations performed in [59]. As already mentioned, the authors of that article were able to solve instances having up to

[11] By way of example, let us consider the solution times obtained for $\varepsilon = 0.05$ in Category (B). Obviously, any of the eight setups was able to solve a total number of 18 (out of 20) instances. Having a look at the average times reported in the table, then it becomes obvious that all of them are very close to 60 seconds which is exactly the contribution of the two instances attempted without success.

ε	Init.	(GLO)	(OK1)	(OK2)	(CPP)
Category (A)					
0.01	✗	39.805 (19)	43.628 (19)	**8.604**	25.365
	✓	12.277	22.515	61.880 (19)	36.870 (19)
0.05	✗	121.85 (17)	129.398 (16)	144.573 (16)	126.919 (17)
	✓	125.953 (16)	124.053 (16)	128.695 (16)	**119.501 (17)**
0.10	✗	30.202	38.313 (19)	30.840 (19)	**29.511**
	✓	30.196 (19)	51.042 (19)	31.316 (19)	33.991 (19)
0.20	✗	0.696	1.484	7.074	**0.435**
	✓	2.650	6.104	0.551	1.477
Category (B)					
0.01	✗	30.291 (19)	**4.844**	31.064 (19)	30.302 (19)
	✓	13.134	25.103	30.316	10.669
0.05	✗	62.546 (18)	61.336 (18)	64.369 (18)	62.959 (18)
	✓	60.874 (18)	66.192 (18)	64.080 (18)	**60.269 (18)**
0.10	✗	56.140 (19)	56.140 (19)	49.710 (19)	**30.373 (19)**
	✓	32.227 (19)	47.120 (19)	34.966 (19)	30.378 (19)
0.20	✗	0.041	0.047	0.096	0.022
	✓	0.007	**0.004**	0.006	0.006

Table 9: Numerical results for the most promising linearizations applied to instances with $n = 20$ items

$n = 18$ items in more than 10 minutes (on average) although using the rather favorable error bound of $\varepsilon = 0.25$. To this end, we now investigate the size of instances that can be solved with the probably best setup we obtained so far. More precisely, we only consider the linearization (CPP) together with a warm start setting and a time limit of 300 seconds, the results of which can be found in Tab. 10. Even for these more challenging instance sizes we see that most of the instances can be solved to proven optimality. Having in mind that an unsolved instance contributes with $300/20 = 15$ seconds to an average value, we further notice that the solution times of the instances successfully attempted are very small, in general. On the other hand, as indicated by the counter *opt*, solving an instance to proven optimality becomes more and more difficult (on average) when n increases, especially for the typically harder sets from Category (B). Nevertheless, Tab. 10 also underlines the benefits of the various approaches presented in this article, as we are able to solve instances larger than those reported in [59] in much less time.

ε	$n = 22$	$n = 24$	$n = 26$	$n = 28$	$n = 30$
Category (A)					
0.01	30.290 (18)	52.368 (17)	58.091 (17)	72.414 (16)	75.732 (15)
0.05	15.139 (19)	45.015 (17)	0.004	31.062 (18)	15.400 (19)
0.10	32.014 (18)	42.108 (18)	0.545	15.225 (19)	41.768 (18)
0.20	15.192 (19)	17.872 (19)	0.279	15.790 (19)	16.049 (19)
opt	**74**	**71**	**77**	**72**	**71**
Category (B)					
0.01	71.265 (16)	61.107 (16)	46.800 (17)	79.217 (15)	42.120 (18)
0.05	4.589	0.035	60.116 (16)	75.114 (15)	105.005 (13)
0.10	0.411	32.832 (18)	16.943 (19)	30.005 (18)	60.451 (16)
0.20	60.079 (16)	30.006 (18)	15.026 (19)	0.004	19.337 (19)
opt	**72**	**72**	**71**	**68**	**66**

Table 10: Numerical results for the most promising setup applied to larger instances

We have already mentioned several times that the absolute distance between the lower and

upper bound values (used in the ILP models) is of great importance for the success of our solution methods, but – in view of the rather small instance sizes considered so far – we have not elaborated on this further. To this end, collecting the data from the 960 instances (with $20 \leq n \leq 30$) contributing to Tab. 9 and Tab. 10, we first state that the absolute difference $\Delta = \min\{ub\} - \max\{lb\}$ between both bounds was either 0 or 1 for every single instance. More precisely, a total number of 109 randomly generated instances (from 480 in total) possessed $\Delta = 1$, and 58 of them (i.e., approximately 63.7%) could be solved to optimality within the given time limit. For the real-world instances, we noted 43 successfully attempted instances (out of 97 with $\Delta = 1$), resulting in a rate of roughly 44.3%. This again underlines the fact that Category (B) typically contains the more challenging instances. A slightly more detailed presentation of these results can be found in Tab. 11, where the precise numbers for both categories are listed for any value of ε.

	Category (A)			Category (B)		
ε	$\Delta = 1$	*opt*	ratio	$\Delta = 1$	*opt*	ratio
0.01	36	18	0.500	36	18	0.500
0.05	25	17	0.680	26	8	0.308
0.10	27	19	0.704	21	11	0.524
0.20	21	17	0.810	14	6	0.430
Σ	109	58	0.637	97	43	0.443

Table 11: Number and ratio of instances with $\Delta = 1$ solved to optimality (depending on ε)

In the majority of the cases, we see that (for both categories) rising the parameter ε leads to a fewer number of cases with $\Delta = 1$. As we already observed earlier, a larger tolerance of overloading the servers is expected to lead to smaller optimal values and, thus, normally smaller values of the bounds, so that the probability of differences (between the lower and upper bounds) is reduced. However, as clearly pointed out in Tab. 6, this is only a very rough explanation since there is no general relationship between ε and the development of the bounds for a fixed instance. At least for Category (A), a further regular structure is related to the ratio of instances solved to optimality. Typically, it should be expected that larger values of ε lead to easier instances, so that they can still be tackled, even if $\Delta = 1$ is satisfied. In terms of the more challenging problems from Category (B), this trend cannot be validated. However, all the instance sets are drawn independently, so that we cannot fully control the total number of hard instances per group of 20 instances used for the tables.

Remark 23. *Very recently, a remarkably easy alternative strategy to deal with the information provided by an upper bound (in the case of minimization problems) has been proposed in [3]. Roughly speaking, when ub is the heuristic value, the cardinality of K (i.e., the number of possibly required servers) can be set to* $ub - 1$ *in advance. If this new problem becomes infeasible, then the heuristic solution was in fact optimal, otherwise we obtain a slightly smaller ILP to solve. This can be particularly relevant in our current experimental setting since we observed that the difference between the best lower and upper bound is at most one for those instances. More precisely, when* $\Delta = 1$ *holds, reducing the heuristic value by one unit would always match the lower bound. To investigate the benefits of this technique we considered all the* 105 *instances with* $\Delta = 1$ *(from Tab. 11) that could not be solved to optimality previously. Note that passing a modified upper bound to the solver also means that we cannot use a feasible starting point anymore, so that we always had to run the cold start mode. Having chosen the same time limit as before, we noticed that roughly* 5% *of the unsolved instances (i.e.,* 5 *out of* 105*) were tackled successfully by the new method. Even if this percentage is rather low, we would consider the methodology itself to be useful. To be more accurate, the optimal value of the vast majority of the instances coincides with the heuristic upper bound ub. Having initialized only* $ub - 1$

servers now, finding a feasible point becomes impossible for the solver so that the branch-and-bound algorithm has to visit (almost) every node before observing infeasibility. Hence, for these instances, the computation times cannot be expected to decrease significantly, in general.

6. Conclusions

In this manuscript, we considered an extension of the classical bin packing problem to normally distributed item sizes. For this optimization problem, being particularly relevant in the context of energy-efficient job-to-server scheduling, we first introduced new lower and upper bounding concepts and extensively discussed their approximation guarantee. As a consequence of these theoretical investigations, in both areas we succeeded in finding approaches that are superior to state-of-the-art techniques from the current literature. Furthermore, these (mathematically) advantageous properties also proved to be beneficial within numerical experiments, covering both randomly generated and real-world instances. Since the SBPP is a nonlinear and non-convex minimization problem, finding an exact solution is very challenging, in general. To facilitate the solution process, we proposed and discussed the application of various potentially helpful strategies including

- five alternative linearization techniques (compared to the original formulation from [59]),
- pre-processing techniques (like fixing variables) based on the information provided by the lower and upper bounds,
- a warm start setup for the commercial solver.

As a main contribution, we saw that the improved bounds form very narrow intervals for the optimal value, thus representing a key element to obtain ILP models easier to handle. On top of that, our numerical results also show that computationally less complex linearization techniques (like (OK1) or (CPP)) additionally boost the solution procedure, in general. Altogether, the methods and results presented in this article significantly extend the range of instances that can be tackled (either exactly or with high-quality approximate solutions) in reasonably short amounts of time.

Regarding future directions of research, we would like to point out two important challenges. On the one hand, the numerical performance of the lower and upper bounds was typically much better than the corresponding approximation results we obtained in our theoretical part. By a much deeper analysis of the applied algorithms, we expect that the constant factors of some approaches can be considerably improved. On the other hand, from a more practical point of view scalability is one of the major issues. So far, any of the currently available exact formulations is of Kantorovich-type, containing some well-known drawbacks. To pave the way for the next stage of growth in the size of instances that can be solved to optimality, alternative modeling frameworks have to be applied. Even if an efficient implementation does not seem to be straightforward, we see high potentials in transferring column generation or flow-based approaches from the BPP to the new stochastic scenario.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Andrae, A.S.G., Edler, T.: On Global Electricity Usage of Communication Technology: Trends to 2030. Challenges 6(1), 117–157 (2015)

[2] Arjona, J., Chatzipapas, A., Fernandez Anta, A., Mancuso, V.: A measurement-based analysis of the energy consumption of data center servers. Proceedings of the 5th International Conference on Future Energy System, 63–74 (2014)

[3] Auer, P., Dósa, G, Dulai, T., Fügenschuh, A., Näser, P., Ortner, R., Werner-Stark, Á.: A new heuristic and an exact approach for a production planning problem. *to appear in:* Central European Journal of Operations Research (`https://doi.org/10.1007/s10100-020-00689-3`) (2020)

[4] Aydin, N., Muter, I., Ilker Birbil, S.: Multi-objective temporal bin packing problem: An application in cloud computing. Computers & Operations Research 121, Article 104959 (2020)

[5] Balas, E.: Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation non linéaire. Technical Report, Comptes rendus de l'Académie des Sciences Paris (1964)

[6] Barnett Jr., T., Jain, S., Andra, U., Khurana, T.: Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017–2022. APJC Cisco Knowledge Network (CKN) Presentation, (*available online:* `https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1213-business-services-ckn.pdf`) (2018)

[7] Belov, G., Scheithauer, G.: A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research 171(1), 85–106 (2006)

[8] Benson, T., Anand, A., Akella, A., Zhang, M.: Understanding data center traffic characteristics. Computer Communication Review 40(1), 92–99 (2010)

[9] Brandão, F., Pedroso, J.P.: Bin packing and related problems: General arc-flow formulation with graph compression. Computers & Operations Research 69, 56–67 (2016)

[10] Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28(5), 755–768 (2012)

[11] Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice & Experience 24(13), 1397–1420 (2012)

[12] Birge, J.R., Louveaux, F.V.: Introduction to stochastic programming. Springer-Verlag, New York (1997)

[13] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)

[14] Chaovalitwongse, W., Pardalos, P.M., Prokopyev, O.A.: A new linearization technique for multi-quadratic 0-1 programming problems. Operations Research Letters 32(6), 517-522 (2004)

[15] Chen, M., Zhang, H., Su, Y.-Y., Wang, X., Jiang, G., Yoshihira, K.: Effective VM sizing in virtualized data centers. Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management and Workshops, 594–601 (2011)

[16] Clautiaux, F., Hanafi, S., Macedo, R., Voge, M.-A., Alves, C.: Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. European Journal of Operational Research 258(2), 467–477 (2017)

[17] Coffman, E.G., Courcoubetis, C., Garey, M.R., Johnson, D.S., McGeoch, L.A., Shor, P.W., Weber, R.R., Yannakakis, M.: Fundamental Discrepancies between Average-Case Analyses under Discrete and Continuous Distributions: A Bin Packing Case Study. Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, 230–240 (1991)

[18] Coffman Jr., E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Survey and classification. In Pardalos, P.M., Du, D., Graham, R.L. (eds), Handbook of Combinatorial Optimization, 455–531, Springer, New York (2013)

[19] Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: An Application of Bin Packing to Multiserver Scheduling. SIAM Journal on Computing 7(1), 1–17 (1978)

[20] Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation Algorithms for Bin Packing – An Updated Survey. In: Ausiello, G., Lucertini, M., Serafini, P. (eds), Algorithm Design for Computer System Design. International Centre for Mechanical Sciences (Courses and Lectures), vol. 284, Springer, Vienna (1984)

[21] Cohen, M.C., Keller, P.W., Mirrokni, V., Zadimoghaddam, M.: Overcommitment in Cloud Services: Bin Packing with Chance Constraints. Management Science 65(7), 3255–3271 (2019)

[22] Crainic, T.G., Gobbato, L., Perboli, G., Rei, W.: Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. European Journal of Operational Research 253(2), 404–417 (2016)

[23] Dargie, W.: A stochastic model for estimating the power consumption of a server. IEEE Transactions on Computers 64(5), 1311–1322 (2015)

[24] de Cauwer, M., Mehta, D., O'Sullivan, B.: The Temporal Bin Packing Problem: An Application to Workload Management in Data Centres. Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence, 157–164, (2016)

[25] Dell'Amico, M., Furini, F., Iori, M.: A Branch-and-Price Algorithm for the Temporal Bin Packing Problem. Computers & Operations Research 114, Article 104825 (2020)

[26] Delorme, M., Iori, M.: Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. INFORMS Journal on Computing 32(1), 101–119 (2020)

[27] Delorme, M. Iori, M., Martello, S.: Bin packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms. Research Report OR-15-1, University of Bologna (2015)

[28] Delorme, M. Iori, M., Martello, S.: Bin packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms. European Journal of Operational Research 255, 1–20 (2016)

[29] Delorme, M. Iori, M., Martello, S.: BPPLIB: a library for bin packing and cutting stock problems. Optimization Letters 12, 235–250 (2018)

[30] Dósa, G., Li, R., Han, X., Tuza, Z.: Tight absolute bound for First Fit Decreasing bin packing: $FFD(L) \leq 11/9 \cdot OPT(L) + 6/9$. Theoretical Computer Science 510, 13–61 (2013)

[31] Dyckhoff, H.: A New Linear Approach to the Cutting Stock Problem. Operations Research 29(6), 1092–1104 (1981)

[32] Fettweis, G., Dörpinghaus, M., Castrillon, J., Kumar, A., Baier, C., Bock, K., Ellinger, F., Fery, A., Fitzek, F., Härtig, H., Jamshidi, K., Kissinger, T., Lehner, W., Mertig, M., Nagel, W., Nguyen, G.T., Plettemeier, D., Schröter, M., Strufe, T.: Architecture and advanced electronics pathways towards highly adaptive energy-efficient computing. Proceedings of the IEEE 107(1), 204–231 (2019)

[33] Fortet, R.: L'algèbre de Boole et ses applications en recherche opérationnelle. Trabajos de Estadistica 11, 111–118 (1960)

[34] Fügenschuh, A.R., Craparo, E.M., Karatas, M., Buttrey, S.M.: Solving multistatic sonar location problems with mixed-integer programming. Optimization and Engineering 21, 273–303 (2020)

[35] Furini, F., Traversi, E.: Theoretical and computational study of several linearisation techniques for binary quadratic problems. Annals of Operations Research 279, 387–411 (2019)

[36] Gilmore, P.C., Gomory, R.E.: A Linear programming approach to the cutting-stock problem (Part I). Operations Research 9, 849–859 (1961)

[37] Glover, F.: Improved linear integer programming formulations of nonlinear integer problems. Management Science 22(4), 455–460 (1975)

[38] Glover, F., Woolsey, E.: Converting the 0-1 polynomial programming problem to a 0-1 linear program. Operations Research 22(1), 180–182 (1974)

[39] Goel, A., Indyk, P.: Stochastic Load Balancing and Related Problems. Proceedings of the 40th Annual Symposium on Foundations of Computer Science, 579–586 (1999)

[40] Gschwind, T., Irnich, S.: Dual Inequalities for Stabilized Column Generation Revisited. INFORMS Journal on Computing 28(1), 175–194 (2016)

[41] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual. *available online:* `http://www.gurobi.com` (2020)

[42] Hähnel, M., Martinovic, J., Scheithauer, G., Fischer, A., Schill, A., Dargie, W.: Extending the Cutting Stock Problem for Consolidating Services with Stochastic Workloads. IEEE Transactions on Parallel and Distributed Systems 29(11), 2478–2488 (2018)

[43] Jin, H., Pan, D., Xu, J., Pissinou, N.: Efficient VM placement with multiple deterministic and stochastic resources in data centers. Proceedings of the IEEE Global Communications Conference, 2505–2510 (2012)

[44] 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 Journal on Computing 3(4), 299-325 (1974)

[45] Jones, N.: How to stop data centres from gobbling up the world's electricity. Nature 561, 163–166 (2018)

[46] Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The Nature of Datacenter Traffic: Measurements & Analysis. Association for Computing Machinery, Internet Measurement Conference (2009)

[47] Kantorovich, L.V.: Mathematical methods of organising and planning production. Management Science 6, 366–422 (1939 Russian, 1960 English)

[48] Kaplan, J.M., Forrest, W., Kindler, N.: Revolutionizing data center energy efficiency. Technical report, McKinsey & Company, (2008)

[49] Keller, G., Tighe, M., Lutfiyya, H., Bauer, M.: An analysis of first fit heuristics for the virtual machine relocation problem. Proceedings of the 8th International Conference on Network and Service Management and Workshop on Systems Virtualization Management, 406–413 (2012)

[50] Kleinberg, J., Rabani, Y., Tardos, E.: Allocating Bandwidth for Bursty Connections. SIAM Journal on Computing 30(1), 191–217 (2000)

[51] Klopfenstein, O., Nace, D.: A robust approach to the chance-constrained knapsack problem. Operations Research Letters 36, 628–632 (2008)

[52] Koomey, J.: Worldwide electricity used in data centers, Environmental Research Letters 3, 1–8 (2008)

[53] Kosuch, S., Lisser,A.: On two-stage stochastic knapsack problems. Discrete Applied Mathematics 159, 1827–1841 (2011)

[54] López-Pires, F., Barán, B.: Virtual Machine Placement Literature Review. arXiv Preprint (*available online:* `http://arxiv.org/abs/1506.01509`) (2015)

[55] Lueker, G.S.: Bin packing with items uniformly distributed over intervals $[a, b]$. Proceedings of the 24th Annual Symposium on Foundations of Computer Science, 289–297 (1983)

[56] Luo, J.-P., Li, X., Chen, M.-R.: Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. Expert Systems with Applications 41(13), 5804–5816 (2014)

[57] Manvi, S.S., Krishna Shyam, G.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. Journal of Network and Computer Applications 41, 424–440 (2014)

[58] Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., Strasdat, N.: Improved flow-based formulations for the skiving stock problem. Computers & Operations Research 113, Article 104770 (2020)

[59] Martinovic, J., Hähnel, M., Scheithauer, G., Dargie, W., Fischer, A.: Cutting Stock Problems with Nondeterministic Item Lengths: A New Approach to Server Consolidation. 4OR 17(2), 173–200 (2019)

[60] Martinovic, J., Scheithauer, G., Valério de Carvalho, J.M.: A Comparative Study of the Arcflow Model and the One-Cut Model for one-dimensional Cutting Stock Problems. European Journal of Operational Research 266(2), 458–471 (2018)

[61] Möbius, C., Dargie, W., Schill, A.: Power consumption estimation models for servers, virtual machines, and servers. IEEE Transactions on Parallel and Distributed Systems 25(6), 1600–1614 (2014)

[62] Monshizadeh Naeen, H., Zeinali, E., Toroghi Haghighat, A.: A stochastic process-based server consolidation approach for dynamic workloads in cloud data centers. Journal of Supercomputing 76(3), 1903–1930 (2020)

[63] Oral, M., Kettani, O.: A linearization procedure for quadratic and cubic mixed-integer problems. Operations Research 40(1), 109–116 (1992)

[64] Padberg, M.: The Boolean quadric polytope: some characteristics, facets and relatives. Mathematical Programming 45, 139–172 (1989)

[65] Perboli, G., Tadei, R., Baldi, M.: The Stochastic Generalized Bin Packing Problem. Discrete Applied Mathematics 160(7-8), 1291–1297 (2012)

[66] Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA (2011)

[67] Rhee, W.T.: Optimal Bin Packing with Items of Random Sizes. Mathematics of Operations Research 13(1), 140–151 (1988)

[68] Rhee, W.T., Talagrand, M.: Optimal Bin Packing with Items of Random Sizes II. SIAM Journal on Computing 18(1), 139–151 (1989)

[69] Scheithauer, G.: Introduction to Cutting and Packing Optimization – Problems, Modeling Approaches, Solution Methods. International Series in Operations Research & Management Science 263, Springer, 1.Edition (2018)

[70] Speitkamp, B., Bichler, M.: A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. IEEE Transactions on Services Computing 3(4), 266–278 (2010)

[71] Tadei, R., Perboli, G.: The Generalized Bin Packing Problem under uncertainty. Proceedings of the 2011 International Conference on Applied & Computational Mathematics, 163–168 (2011)

[72] Talebian, H., Gani, A., Sookhak, M., Abdelatif, A.A., Yousafzai, A., Vasilakos, A.V., Yu, F.R.: Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues. Cluster Computing 23, 837–878 (2020)

[73] Tighe, M., Keller, G., Bauer, M., Lutfiyya, H.: A distributed approach to dynamic VM management. Proceedings of the 9th International Conference on Network and Service Management, 166–170 (2013)

[74] Usmani, Z., Singh, S.: A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. Procedia Computer Science 78, 491–498 (2016)

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

[76] Valério de Carvalho, J.M.: LP models for bin packing and cutting stock problems. European Journal of Operations Research 141(2), 253–273 (2002)

[77] Valério de Carvalho, J.M.: Using Extra Dual Cuts to Accelerate Column Generation. INFORMS Journal on Computing 17(2), 175–182 (2005)

[78] Vance, P.: Branch-and-price algorithms for the one-dimensional cutting stock problem. Computational Optimization and Applications 9, 211–228 (1998)

[79] Vance, P., Barnhart, C., Johnson, E.L., Nemhauser, G.L.: Solving binary cutting stock problems by column generation and branch-and-bound. Computational Optimization and Applications 3(2), 111–130 (1994)

[80] Wang, M., Meng, X., Zhang, L.: Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data Centers. Proceedings of the IEEE INFOCOM, 71–75 (2011)

[81] Watters, L.J.: Letter to the editor: Reduction of integer polynomial programming problems to zero-one linear programming problems. Operations Research 6(15), 1171–1174 (1967)

[82] Wei, L., Luo, Z., Baldacci, R., Lim, A.: A new branch-and-price-and-cut algorithm for one-dimensional bin packing problems. INFORMS Journal on Computing 32(2), 428–443 (2020)

[83] Wu, Y.: Energy efficient virtual machine placement in data centers. Master thesis, Queensland University of Technology (2013)

[84] Yu, L., Chen, L., Cai, Z., Shen, H., Liang, Y., Pan, Y.: Stochastic Load Balancing for Virtual Resource Management in Datacenters. IEEE Transactions on Cloud Computing 8(2), 459–472 (2020)

[85] Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., Chao, K.-M., Li, J.: Virtual machine consolidated placement based on multi-objective biogeography-based optimization. Future Generation Computer Systems 54, 95–122 (2016)

AppendixA. Theoretical Background for Theorem 12

Let us start with an auxiliary result which will be used in the proof of our main theorem:

Lemma 24. *Let real numbers $A, B, C \in \mathbb{R}$ with $A \geq B \geq C > 0$ be given. Then we have*

$$\sqrt{A+C} + \sqrt{B-C} - \sqrt{A} < \sqrt{B}. \tag{A.1}$$

Proof. Based on the given assumptions, we first observe that $C \cdot (B-A) - C^2 < 0$ has to hold. Adding $A \cdot B$ on both sides of this inequality leads to $(A+C) \cdot (B-C) < AB$. Due to $B \geq C$, the left hand side is nonnegative (and so is the right hand side). Consequently, we obtain

$$\sqrt{A+C} \cdot \sqrt{B-C} < \sqrt{AB}.$$

Multiplying both sides by 2 and adding the term $A+B$ afterwards leads to

$$A + 2 \cdot \sqrt{A+C} \cdot \sqrt{B-C} + B < A + 2 \cdot \sqrt{AB} + B.$$

By means of a little trick, this inequality can be written as

$$(A+C) + 2 \cdot \sqrt{A+C} \cdot \sqrt{B-C} + (B-C) < A + 2 \cdot \sqrt{AB} + B$$

which entails

$$\sqrt{A+C} + \sqrt{B-C} < \sqrt{A} + \sqrt{B}$$

thanks to the assumptions of this lemma. Finally, we end up with the claim by rearranging the terms. □

Before proceeding with the theoretical consideration, two special types of solutions shall be highlighted:

Definition 3. *Let E be an instance of the SBPP, and let z denote the optimal value of $\Phi(E)$. An optimal solution (x, y) of $\Phi(E)$ is called perfect, if only the first z bins are active and if (16) holds with equality for $k = 1, \ldots, z-1$, i.e., all bins (possibly except for the very last active bin) are packed without any unused space. The set of all perfect solutions will be referred to as $S^{\Phi}_{\text{perf}}(E)$.*

Obviously, any optimal solution (x, y) of $\Phi(E)$ can be transferred to a corresponding perfect solution by iteratively shifting (fractions of) items from bin k to bin $k-1$, if bin $k-1$ is not completely filled. Hence, considering perfect solutions does not represent an actual restriction.

Definition 4. *Let E be an instance of the SBPP, and let z denote the optimal value of $\Phi(E)$. An optimal solution (x, y) of $\Phi(E)$ is called ordered, if only the first z bins are active and if the implication*

$$x_{ik} > 0 \Longrightarrow x_{j,k+1} = 0 \tag{A.2}$$

holds for any $k \in \{1, \ldots, z-1\}$ and any $i, j \in I$ with $\sigma_j^2/\mu_j > \sigma_i^2/\mu_i$. The set of all ordered solutions will be referred to as $S^{\Phi}_{\text{ord}}(E)$.

In an ordered solution, the items are packed into the bins in a successive way, following a non-increasing order of the fractions σ_i^2/μ_i. In particular, we have:

Remark 25. *For any perfect solution of $\Phi(E)$, Condition (A.2) is equivalent to the property that*

$$x_{ik} > 0 \Longrightarrow x_{jl} = 0 \tag{A.3}$$

holds for any $k, l \in \{1, \ldots, z\}$ with $l > k$ and any $i, j \in I$ with $\sigma_j^2/\mu_j > \sigma_i^2/\mu_i$. While it is clear that for $l = k + 1$ Condition (A.2) is implied by (A.3), the reverse direction can be verified by an iterated application of (A.2) in the following manner: By way of example, let $x_{ik} > 0$ and $x_{j,k+2} > 0$ hold for some $i, j \in I$ with

$$\frac{\sigma_j^2}{\mu_j} > \frac{\sigma_i^2}{\mu_i}. \tag{A.4}$$

Due to (A.2), all items $p \in I$ in bin $k + 1$ satisfy

$$\frac{\sigma_p^2}{\mu_p} \leq \frac{\sigma_i^2}{\mu_i}. \tag{A.5}$$

Using (A.2) again for bin $k + 1$ now leads to the fact, that there is no item $q \in I$ in bin $k + 2$ with $\sigma_q^2/\mu_q > \sigma_p^2/\mu_p$. But remember that, by hypothesis, we have $q = j$ in bin $k + 2$ leading to the following contradiction

$$\frac{\sigma_q^2}{\mu_q} = \frac{\sigma_j^2}{\mu_j} \overset{(A.4)}{>} \frac{\sigma_i^2}{\mu_i} \overset{(A.5)}{\geq} \frac{\sigma_p^2}{\mu_p}.$$

In a similar way, the proof can be done for arbitrary $l > k$.

The next contribution shows that there is a perfect solution of $\Phi(E)$ that is ordered, too.

Theorem 26. *We have $S^{\Phi}_{\text{perf}}(E) \cap S^{\Phi}_{\text{ord}}(E) \neq \emptyset$.*

Proof. Let z denote the optimal value of $\Phi(E)$, and let us assume that $K = \{1, \ldots, z\}$ holds for simplicity (i.e., we do not consider additional empty bins). For the sake of contradiction, let us assume that the claim is not true. As already mentioned, we cannot have $S^{\Phi}_{\text{perf}}(E) = \emptyset$, and so we can actually choose the perfect solution $(x, y) \in S^{\Phi}_{\text{perf}}(E)$ having the lexicographically largest vector

$$\left(\sum_{i \in I} x_{i1}\sigma_i^2, \sum_{i \in I} x_{i2}\sigma_i^2 \ \ldots, \sum_{i \in I} x_{iz}\sigma_i^2 \right). \tag{A.6}$$

Since it is possible to permute the items between the bins, note that this particularly implies

$$\sum_{i \in I} x_{i1}\sigma_i^2 \geq \sum_{i \in I} x_{i2}\sigma_i^2 \geq \ldots \geq \sum_{i \in I} x_{iz}\sigma_i^2. \tag{A.7}$$

Because of $(x, y) \notin S^{\Phi}_{\text{ord}}(E)$, there is some $k \in \{1, \ldots, z - 1\}$ and $i, j \in I$ with

$$x_{ik} > 0, \qquad x_{j,k+1} > 0, \qquad \frac{\sigma_j^2}{\mu_j} > \frac{\sigma_i^2}{\mu_i}. \tag{A.8}$$

Step 1: Constructing a candidate
Let us consider

$$\rho := \min \left\{ x_{ik}, x_{j,k+1} \cdot \frac{\mu_j}{\mu_i} \right\}, \tag{A.9}$$

and let us start defining an alternative solution $(x^\star, y^\star)$ of $\Phi(E)$ by $x^\star_{ik} := x_{ik} - \rho$ and $x^\star_{i,k+1} := x_{i,k+1} + \rho$. As we decreased the fraction of item i in bin k a bit, this bin is not completely filled anymore. Hence, we add a small portion of item j to bin k by means of $x^\star_{jk} := x_{jk} + \tau$ with

$$\tau := \max \left\{ t \geq 0 \ : \ \sum_{i \in I} x_{ik}\mu_i - \rho \cdot \mu_i + t \cdot \mu_j + q_{1-\varepsilon} \cdot \sqrt{\sum_{i \in I} x_{ik}\sigma_i^2 - \rho\sigma_i^2 + t\sigma_j^2} \leq 1 \right\}, \tag{A.10}$$

i.e., τ is defined in such a way, that Condition (16) is satisfied with equality also for the components of $x^\star$ belonging to bin k. Consequently, we need to decrease the fraction of item j appearing in bin $k+1$ in the same way, leading to $x^\star_{j,k+1} := x_{j,k+1} - \tau$. All other components of $x^\star$ and $y^\star$ stay the same as in the original solution (x, y).

Step 2: Feasibility check
Now, let us consider two cases:

- **Case 1:** $\tau\mu_j \geq \rho\mu_i$
  In this case, the capacity constraint of bin k is violated due to the following calculation:

$$\begin{aligned}
& \sum_{i\in I} x^\star_{ik}\mu_i + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x^\star_{ik}\sigma_i^2} \\
= \; & \sum_{i\in I} x_{ik}\mu_i + \tau\mu_j - \rho\mu_i + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 + \tau\sigma_j^2 - \rho\sigma_i^2} \\
\overset{(A.8)}{>} \; & \sum_{i\in I} x_{ik}\mu_i + \tau\mu_j - \rho\mu_i + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 + \tau\cdot\sigma_i^2\cdot\frac{\mu_j}{\mu_i} - \rho\sigma_i^2} \\
= \; & \sum_{i\in I} x_{ik}\mu_i + \underbrace{\tau\mu_j - \rho\mu_i}_{\geq 0} + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 + \frac{\sigma_i^2}{\mu_i}\cdot\underbrace{(\tau\mu_j - \rho\mu_i)}_{\geq 0}} \\
\geq \; & \sum_{i\in I} x_{ik}\mu_i + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2} = 1.
\end{aligned}$$

  Hence, this case cannot occur.

- **Case 2:** $\tau\mu_j < \rho\mu_i$
  First of all, we prove that the new candidate $(x^\star, y^\star)$ is feasible (and thus optimal, since the number of required bins did not change). To this end, it is sufficient to concentrate on those conditions that involve variables which actually have changed (compared to the original point (x, y)). This leads to the following observations:

  - We have
$$x^\star_{ik} = x_{ik} - \rho \overset{(A.9)}{\geq} x_{ik} - x_{ik} = 0,$$
    i.e., Condition (18) is true for (i, k).
  - We have $x^\star_{i,k+1} = x_{i,k+1} + \rho \geq x_{i,k+1} \geq 0$ and $x^\star_{jk} = x_{jk} + \tau \geq x_{jk} \geq 0$, i.e., Condition (18) is true for $(i, k+1)$ and (j, k).
  - By means of the hypothesis of the current case, we obtain
$$x^\star_{j,k+1} = x_{j,k+1} - \tau > x_{j,k+1} - \rho\cdot\frac{\mu_i}{\mu_j} \overset{(A.9)}{\geq} x_{j,k+1} - x_{j,k+1}\cdot\frac{\mu_j}{\mu_i}\cdot\frac{\mu_i}{\mu_j} = 0,$$
    i.e., Condition (18) is true for $(j, k+1)$.
  - Obviously, Conditions (15) are satisfied for items i and j, since (in each case) one corresponding variable is increased by exactly the same amount which is subtracted from another variable.
  - By construction of τ, see (A.10), we know that Condition (16) holds with equality for bin k.
  - The only thing which remains to be checked now is Condition (16) for bin $k+1$. As this requires some further preparation, we refer the reader to Step 4 of this proof.

Step 3: Two auxiliary results
As we observed, bin k satisfies Condition (16) with equality for both considered points, i.e., for (x, y) and for $(x^\star, y^\star)$. Hence, we have

$$\begin{aligned} 1 &= \sum_{i\in I} x_{ik}\mu_i + q_{1-\varepsilon}\cdot\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2} \\ 1 &= \sum_{i\in I} x^\star_{ik}\mu_i + q_{1-\varepsilon}\cdot\sqrt{\sum_{i\in I} x^\star_{ik}\sigma_i^2} \\ &= \sum_{i\in I} x_{ik}\mu_i - \rho\mu_i + \tau\mu_j + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 - \rho\sigma_i^2 + \tau\sigma_j^2} \end{aligned}$$

In particular, we obtain the following two observations:

i) Since we are still in the second case of Step 2, $-\rho\mu_i + \tau\mu_j < 0$ is true. This leads to

$$\sum_{i\in I} x^\star_{ik}\mu_i < \sum_{i\in I} x_{ik}\mu_i \qquad \text{and} \qquad \sum_{i\in I} x^\star_{ik}\sigma_i^2 > \sum_{i\in I} x_{ik}\sigma_i^2.$$

In particular, the latter also implies $-\rho\sigma_i^2 + \tau\sigma_j^2 > 0$.

ii) By subtracting the first line from the last line (in the system of equations at the beginning of Step 3), we obtain

$$\begin{aligned} -\rho\mu_i + \tau\mu_j + q_{1-\varepsilon}\cdot\left(\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 - \rho\sigma_i^2 + \tau\sigma_j^2} - \sqrt{\sum_{i\in I} x_{ik}\sigma_i^2}\right) &= 0, \\ \iff q_{1-\varepsilon}\cdot\left(\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 - \rho\sigma_i^2 + \tau\sigma_j^2} - \sqrt{\sum_{i\in I} x_{ik}\sigma_i^2}\right) &= \rho\mu_i - \tau\mu_j. \end{aligned}$$

Step 4: Finishing the feasibility check from Step 2
For bin $k+1$, we now obtain

$$\begin{aligned} & \sum_{i\in I} x^\star_{i,k+1}\mu_i + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x^\star_{i,k+1}\sigma_i^2} \\ = \quad & \sum_{i\in I} x_{i,k+1}\mu_i + \rho\mu_i - \tau\mu_j + q_{1-\varepsilon}\sqrt{\sum_{i\in I} x_{i,k+1}\sigma_i^2 + \rho\sigma_i^2 - \tau\sigma_j^2} \\ \overset{\text{Step 3,}ii)}{=} \quad & \sum_{i\in I} x_{i,k+1}\mu_i + q_{1-\varepsilon}\cdot\left[\sqrt{\sum_{i\in I} x_{ik}\sigma_i^2 - \rho\sigma_i^2 + \tau\sigma_j^2} + \sqrt{\sum_{i\in I} x_{i,k+1}\sigma_i^2 + \rho\sigma_i^2 - \tau\sigma_j^2}\right. \\ & \left. - \sqrt{\sum_{i\in I} x_{ik}\sigma_i^2}\right] \\ \overset{\text{Lemma 24}}{\leq} \quad & \sum_{i\in I} x_{i,k+1}\mu_i + q_{1-\varepsilon}\cdot\sqrt{\sum_{i\in I} x_{i,k+1}\sigma_i^2} \leq 1. \end{aligned}$$

As regards the application of Lemma 24, we have to choose

$$A := \sum_{i\in I} x_{ik}\sigma_i^2, \quad B := \sum_{i\in I} x_{i,k+1}\sigma_i^2, \quad C := -\rho\sigma_i^2 + \tau\sigma_j^2.$$

Then, the requirements of the lemma are satisfied: Indeed, we have $A \geq B$ due to (A.7), $B \geq C$ since $\sum_{i \in I} x^{\star}_{i,k+1} \sigma_i^2 \geq 0$ has to hold and $C > 0$ as a consequence of Step 3, i).

Altogether, also bin $k+1$ does respect the capacity constraint, so that $(x^{\star}, y^{\star})$ is feasible (and also optimal). Our calculation did not answer, whether bin $k+1$ is actually completely filled for this new solution. However, even in the case that equality does not hold in the capacity constraint (16), we can iteratively shift items forward from the higher-indexed bins (if any), to obtain again a perfect solution. (Of course, if $k+1 = z$ is the last bin, then it does not have to be perfectly filled.)

Step 5: The contradiction
Now we have built a new perfect solution (for simplicity, also referred to as $(x^{\star}, y^{\star})$). Moreover, given the observation in Step 3, i), this new solution satisfies

$$\sum_{i \in I} x^{\star}_{ik} \sigma_i^2 > \sum_{i \in I} x_{ik} \sigma_i^2.$$

As we did not touch bins with indices smaller than k, the solution $(x^{\star}, y^{\star})$ actually represents a perfect solution whose vector

$$\left(\sum_{i \in I} x^{\star}_{i1} \sigma_i^2, \sum_{i \in I} x^{\star}_{i2} \sigma_i^2 \ \ldots, \sum_{i \in I} x^{\star}_{iz} \sigma_i^2 \right)$$

is lexicographically larger than that corresponding to the original solution (x, y). This contradicts to the initial assumption also making Case 2 impossible to happen. Altogether, our initial assumption has to be wrong, and we have $S^{\Phi}_{\text{perf}}(E) \cap S^{\Phi}_{\text{ord}}(E) \neq \emptyset$, i.e., there is a perfect and ordered solution of $\Phi(E)$. □

Remark 27. *Note that we need $q_{1-\varepsilon} > 0$ in this proof (e.g., in Case 1 of Step 2), so these considerations only hold for $0 \leq \varepsilon < 1/2$.*

At last, we notice that an optimal solution having both required properties is precisely computed by the NFD heuristic presented in Alg. 1.