# Asymptotic fully polynomial approximation schemes for variants of open-end bin packing

Leah Epstein* Asaf Levin †

**Abstract**

We consider three variants of the open-end bin packing problem. Such variants of bin packing allow the total size of items packed into a bin to exceed the capacity of a bin, provided that a removal of the last item assigned to a bin would bring the contents of the bin below the capacity. In the first variant, this last item is the minimum sized item in the bin, that is, each bin must satisfy the property that the removal of any item should bring the total size of items in the bin below 1. The next variant (which is also known as *lazy bin covering* is similar to the first one, but in addition to the first condition, all bins (expect for possibly one bin) must contain a total size of items of at least 1. We show that these two problems admit asymptotic fully polynomial time approximation schemes (AFPTAS). Moreover, they turn out to be equivalent. We briefly discuss a third variant, where the input items are totally ordered, and the removal of the maximum indexed item should bring the total size of items in the bin below 1, and show that this variant is strongly NP-hard.

## 1 Introduction

In bin packing problems, the input is a set of items that are to be partitioned into subsets called *bins*, with the goal of minimizing the number of subsets, according to some restriction on the contents of one subset. Specifically, the input consists of a set of $n$ items $I = \{1, 2, \ldots, n\}$, where the size of item $i$ is $s_i \in [0, 1]$. The goal of the STRONG OPEN-END BIN PACKING PROBLEM (SOBP) is to partition the items into a minimum number of subsets, such that for any such subset $S$, $\sum_{i \in S} s_i - \min_{i \in S} s_i < 1$. That is, removing any item from $S$ results in a total sum of less than 1. The goal of the ORDERED OPEN-END BIN PACKING PROBLEM (OOBP) is to partition the items into a minimum number of subsets, such that for any such subset $S = \{i_1, i_2, \ldots, i_k\}$, where $i_1 < i_2 < \ldots < i_k$, $\sum_{i \in S} s_i - s_{i_k} < 1$. That is, removing the item with the maximum index from $S$ (also called the *last item*) results in a total sum of less than 1. The goal of the LAZY BIN COVERING (LBC) problem is to partition the items into a minimum number of subsets, such that for any such subset $S$, $\sum_{i \in S} s_i - \min_{i \in S} s_i < 1$, as in SOBP, and in addition, all subsets (except at most one) must satisfy $\sum_{i \in S} s_i \geq 1$. That is, the goal is to minimize the number of covered bins, where a bin needs to contain a minimum set of items (with respect to containment) that has a total size of at least 1. One bin may remain uncovered.

*Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

†Department of Statistics, The Hebrew University, Jerusalem, Israel. levinas@mscc.huji.ac.il.

Consider the following situation. A group of workers needs to be hired to complete a set of tasks of given durations. The employees start working at 8 a.m. and can leave starting from 4 p.m. (but may possibly need to stay later). Each worker needs to process the tasks assigned to him continuously, one by one during the day, without breaks. The worker can go home no earlier than 4 p.m. and only after he completed all tasks that he started (so if there is a task that the worker did not start by 4 p.m., then he can leave it undone). The goal of the boss is to partition the tasks in a way such that the number of workers required to complete all tasks in one day is minimized. If the worker is free to decide on the order in which he processes the tasks, we get SOBP. If a total order is given on the tasks, and each worker must process the tasks according to this order, then we get OOBP. If the employer tries to assign the tasks so that at most one worker can be ever idle for some time (after finishing the tasks and before going home), then we get LBC. Applications of open-end packing problems were presented in previous work [12, 8, 13].

For an algorithm $\mathcal{A}$, we denote its cost on an input $X$ by $\mathcal{A}(X)$, and if $X$ is clear from the context, we simply use $\mathcal{A}$. An optimal algorithm is denoted by OPT. For minimization problems, the asymptotic approximation ratio of an algorithm $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input $X$, $\mathcal{A}(X) \leq \mathcal{R} \cdot \text{OPT}(X) + c$, where $c$ is a constant independent of the input. If we enforce $c = 0$, $\mathcal{R}$ is called the absolute approximation ratio.

An asymptotic polynomial time approximation scheme is a family of approximation algorithms, such that for every $\varepsilon > 0$ the family contains a polynomial time algorithm with an asymptotic approximation ratio of $1 + \varepsilon$. We abbreviate *asymptotic polynomial time approximation scheme* by APTAS (also called an asymptotic PTAS). An asymptotic fully polynomial time approximation scheme (AFPTAS) is an APTAS whose time complexity is polynomial not only in the input size but also in $\frac{1}{\varepsilon}$. The constant $c$ in the definition above may depend on $\frac{1}{\varepsilon}$. If the scheme satisfies the definition with $c = 0$, stronger results are obtained, namely, polynomial time approximation schemes and fully polynomial approximation schemes, which are abbreviated as PTAS and FPTAS. It is known that the standard bin packing problem admits an APTAS [4] and an AFPTAS [7].

Several variants of open-end packing problems were introduced and studied in [12, 8, 13, 10, 9, 5]. Leung, Dror and Young [8] and Zhang [13] studied a fourth variant of open-end bin packing, in which the removal of the last item brings the bin to a level of less than 1, but the items can be packed into the bin in an order which is chosen by the algorithm, that is, the goal is to partition the items into the minimum number of subsets, such that for any such subset $S$, $\sum_{i \in S} s_i - \max_{i \in S} s_i < 1$. It was shown by [12] that this fourth variant is NP-hard and admits an AFPTAS. The AFPTAS uses the AFPTAS for standard bin packing as a subroutine. Specifically, the largest items in the sequence are packed as last items, and the remaining items form an instance of standard bin packing. The application above in the fare payment system is relevant for this variant as well, if the passenger can decide on an order in which the trips are done.

Yang and Leung studied OOBP in an online scenario [12]. The LAZY BIN COVERING was studied by Lin, Yang and Xu [10, 9]. In [10], they claimed that the problem is NP-hard and that it admits an APTAS. Their proof of NP-hardness is via reduction from PARTITION where the total size of the items is $\Theta(b)$ and the number of items created as an instance for LBC is $\Theta(b)$ and hence this proof is inaccurate. We note that if the input to LBC can be given as a set of pairs $(s_i, n_i)$ where $n_i$ is the multiplicity of the items of size $s_i$, then their proof can be corrected. A corrected proof of the NP-hardness and APX-hardness of LBC was recently given by Gai and Zhang [5] (in their terminology our LBC is called RESTRICTED OPEN-END BIN PACKING). Note that the same paper provides NP-hardness and APX-hardness for an additional variant, in which a non-full bin does not count towards the goal function.

The claimed result of the APTAS of [10] is given without details on the scheme, however this result is improved by our results. They further study fast heuristics for the problem. Specifically, FIRST FIT DECREASING (FFD) is defined as follows. The items are sorted by non-increasing size. At each time, the next bin receives a minimum prefix of items of a total size of at least 1. If the remaining items have a

total size of less than 1, they are packed into one last bin. Due to the sorting, the last item in each bin is the minimum sized item, and therefore every created bin is valid. This algorithm can be called NEXT FIT DECREASING (NFD) as well, since the algorithm never returns to pack additional items in previously created bins. It was shown in [10] that the asymptotic approximation ratio of this algorithm is $\frac{71}{60} \approx 1.18333$. In [9], an improved algorithm of an asymptotic approximation ratio of $\frac{17}{15} \approx 1.13333$ was designed. FFD was shown to be also a best possible absolute approximation algorithm for LBC by [5].

Another packing problem in which bins can have a total contents of more than 1, is the BIN COVERING problem [1, 3, 2, 6], which is a maximization problem. In this problem, the goal is to maximize the number of bins which have a total size of items of at least 1. BIN COVERING admits an APTAS [2] and an AFPTAS [6]. Note that the problem is a maximization problem and it is therefore different in nature from the problems studied here.

**Our results.** In Section 2, we show that SOBP and LBC are equivalent, that is, given a solution to one of the problems, which uses $k$ bins, a solution to the other problem, which uses at most $k$ bins, can be created in polynomial time. We further show that OOBP is strongly NP-hard. In Section 3, we design an AFPTAS for SOBP. This immediately results in an AFPTAS for LBC, due to the reduction of Section 2.

## 2 Preliminaries

We start by showing an equivalence between SOBP and LBC. The algorithmic approach allows to convert a solution for one problem to the other problem.

**Theorem 1** *The problems,* SOBP *and* LBC *are polynomially equivalent.*

**Proof.** By definition, any solution for LBC is a valid solution for SOBP. Therefore, we show how to convert a solution for SOBP, which uses $k$ bins, into a solution to LBC, which uses at most $k$ bins. This shows that given a set of items, the costs of an optimal solutions for the two problems are equal.

Given a solution to $SOBP$, bins which contain a total size of items which is at least 1, are unchanged. Consider the set of bins containing items of a total size smaller than 1. Let $j$ be the number of such bins. If $j \leq 1$ then the solution is a valid solution for LBC. Otherwise, remove these items and repack them using FFD. The resulting packing is clearly a valid solution for LBC, since at most one bin would contain a total size of items of less than 1. Moreover, the number of bins used it at most $j$. To show this, assume by contradiction that at least $j+1$ bins are created. The first $j$ bins receive a total size of items of at least $j$ (in all $j$ bins together). However, we assumed that these items were packed in $j$ bins, where each one of these bins had a total size of items which was smaller than 1. This gives a total which is smaller than $j$ and yields a contradiction. ■

We next consider OOBPand show the following.

**Theorem 2** *The problem* OOBP *is NP-hard in the strong sense.*

**Proof.** We consider a reduction from the 3-PARTITION problem. In this problem, the input consists of a number $m$, a number $b$, and $3m$ numbers of sizes $a_i$ (for $1 \leq i \leq 3m$), such that $\frac{b}{4} < a_i < \frac{b}{2}$ for $1 \leq i \leq 3m$, and $\sum_{i=1}^{3} ma_i = mb$. The goal is to partition them into $m$ subsets, with 3 items each, such that the sum of the three items in each subset is $b$.

We create the following input to OOBP. The first $3m$ items have sizes $\frac{2a_i}{2b+1}$. The next $m$ items all have size 1.

We claim that it is possible to partition the $3m$ numbers as required, if and only if the cost of an optimal solution for the input of OOBP is at most $m$.

If it is possible to partition the $3m$ numbers into $m$ subsets, we define a triple of items to be the three items resulting from a triple of numbers of the input of the 3-PARTITION problem. Then it is possible to pack every triple of items in a bin together with one item of size 1. The item of size 1 appears in the sequence later than all smaller items, therefore, since the sum of the triple is $\frac{2b}{2b+1}$, the packing is valid, and uses $m$ bins.

If a solution for OOBP, which uses at most $m$ bins exists, we first note that every bin contains exactly one item of size 1. This holds since two such items cannot be packed in one bin. The sum of other items in each such bin must be below 1, and therefore, their sum is at most $\frac{2b}{2b+1}$. The sum of the first $3m$ items in the sequence is $\frac{2bm}{2b+1}$, and therefore we get that every bin contains a subset of items of sum $\frac{2b}{2b+1}$, which implies a subset of three numbers of total sum $b$. Therefore, we get a valid 3-partition of the numbers. ■

# 3 An AFPTAS for SOBP

Let $\varepsilon \leq 1$. We design an asymptotic fully polynomial time approximation scheme, that is an algorithm of asymptotic approximation ratio $1+\varepsilon$, whose running time is polynomial in the input size and in $\frac{1}{\varepsilon}$. We assume that all items have a strictly positive size, i.e., $s_i > 0$ for all $i$. If the original input contains any zero sized items, these items can be packed in a dedicated bin, adding 1 to the cost of the resulting solution.

We apply linear grouping as in [4]. We define *small* items as items of size at most $\varepsilon$, and all other items (each of which has a size strictly larger than $\varepsilon$) are *large*. These sets are denoted by $T$ and $L$, respectively.

We partition the set of large items $L$ into $\frac{1}{\varepsilon^2}$ classes $L_1, L_2, \ldots, L_{\frac{1}{\varepsilon^2}}$ such that $\lceil |L|\varepsilon^2 \rceil = |L_1| \geq |L_2| \geq \cdots \geq |L_{\frac{1}{\varepsilon^2}}| = \lfloor |L|\varepsilon^2 \rfloor$ (note that this condition uniquely identifies the cardinality of each class), and such that if there are two items $i, j$ with sizes $s_i > s_j$ and $i \in L_q$ and $j \in L_p$ then $q \leq p$ (so $L_1$ receives the subset of largest items, and for $2 \leq p \leq \frac{1}{\varepsilon^2}$, $L_p$ receives the largest items from $L \setminus (L_1 \cup \cdots \cup L_{p-1})$). The two conditions uniquely define the allocation of items into classes up to the allocation of equal sized items.

Then, we round up the sizes of the items in $L_2, \ldots, L_{1/\varepsilon^2}$ as follows: For all values of $p$, $p = 2, 3, \ldots, \frac{1}{\varepsilon^2}$, and for each item $i \in L_p$, we define $s'_i = \max_{j \in L_p} s_j$ to be the *rounded-up size of item* $i$. For any $i \in T$ we let $s'_i = s_i$. The rounded-up instance $I'$ consists of the set of items $T \cup L \setminus L_1$, where for every $i$, the size of item $i$ is $s'_i$.

We define a process of *reduction* on a valid bin, in which every item may be replaced by a smaller item or removed completely. We claim that such a process results in a valid bin as well. To see this note that an alternative definition of the problem is that a set of items is a valid bin if every proper subset of this set has a total size of at most 1. Clearly, this property does not change by a reduction process.

**Claim 3** $\text{OPT}(I') \leq \text{OPT}$

**Proof.** Given an optimal solution to $I$, OPT, we transform it into a solution to $I'$ using a reduction process on bins. We define a bijection from $I'$ to $I$, so that every item of $I'$ is mapped to an item of $I$ which is no smaller, and can take its place in the packing. Since $I'$ does not contain $L_1$, and since $|L_i| \leq |L_{i-1}|$ (in both $I$ and $I'$, since the cardinality of sets is not influenced by the rounding), we map every item of $L_i$ (for all $i \geq 2$) in $I'$ to some item of $L_{i-1}$ in $I$. By our rounding, every item of $L_i$ in $I'$ is no larger than any item of $L_{i-1}$ in $I$. ■

Let $H$ be the set of distinct sizes of items in $L'$. By definition, we have $|H| \leq \frac{1}{\varepsilon^2}$. For a size $v \in H$, we define a partial $v$-configuration as a multi-set of sizes in $H$, that contains only sizes which are no smaller than $v$, and has a total size strictly smaller than 1. The set of such partial configurations is denoted by $P(v)$. A complete $v$-configuration results from a partial $v$-configuration, which is augmented with one additional item of size $v$. The set of such complete configurations is denoted by $\mathcal{C}(v)$. Let $C(\emptyset)$ be an empty

configuration. We let $\mathcal{C} = \bigcup_{v \in H} \mathcal{C}(v) \cup \{C(\emptyset)\}$. For a configuration $C \in \mathcal{C}$, we define the number of items of size $v$ in it by $n(v, C)$, for all $v \in H$. We partition $\mathcal{C}$ into overloaded configurations $\mathcal{C}_1$ and regular configurations $\mathcal{C}_2$. $\mathcal{C}_1$ is the set of all configurations for which the sum of sizes is at least 1, and $\mathcal{C}_2$ is the set of all configurations where the sum of sizes is smaller than 1, that is, $C \in \mathcal{C}$ is in $\mathcal{C}_1$ if $\sum_{v \in H} v \cdot n(v, C) \geq 1$ and $C \in \mathcal{C}$ is in $\mathcal{C}_2$ if $\sum_{v \in H} v \cdot n(v, C) < 1$. The sets $\mathcal{C}$,$\mathcal{C}_1$ and $\mathcal{C}_2$ are not computed explicitly, and typically have exponential sizes.

For $C \in \mathcal{C}_2$, let $\Delta(C)$ be an upper bound on the space in this configuration that can be used by small items , i.e., $\Delta(C) = 1 + \varepsilon - \sum_{v \in H} v \cdot n(v, C)$. This is indeed an upper bound on the total size of small items since the presence of a small item implies that the total size of all items in the bin is at most $1 + \varepsilon$.

For every $v \in H$, we let $n(v)$ be the number of items in $L'$, with size $v$. We let $\Delta = \sum_{i \in T} s'_i$ be the total size of the small items. The following linear program is solved approximately. For each configuration $C \in \mathcal{C}$, there is a variable $x_C$ indicating the number of bins packed using configuration $C$.

$$
\begin{aligned}
\min \quad & \sum_{C \in \mathcal{C}} x_C \\
s.t. \quad & \sum_{C \in \mathcal{C}} n(v, C) x_C \geq n(v) \quad \forall v \in H && (1) \\
& \sum_{C \in \mathcal{C}_2} \Delta(C) x_C \geq \Delta && (2) \\
& x_C \geq 0 \qquad \forall C \in \mathcal{C}.
\end{aligned}
$$

We consider the relation of the linear program to feasible solutions. Specifically, we show that for any feasible packing which uses $b$ bins, there exists a feasible solution to the linear program, for which the value of the objective function is $b$. This implies that the same holds for an optimal solution to the packing problem. Consider a solution to the rounded-up instance, which uses $b$ bins, and the packing of $L'$. We let $\tilde{x}_C$ be the number of bins packed by configuration $C \in \mathcal{C}$. Constraint (1) is therefore satisfied. Consider all bins in the packing that contain at least one small item. Clearly, every such bin is packed using a configuration from $\mathcal{C}_2$. Moreover, such bins may contain a total size of items of at most $1+\varepsilon$. Therefore, the space used by small items satisfies Constraint (2). Hence, the resulting vector $\tilde{x}$ is a feasible solution to the linear program, and its cost is $b$.

We let $\hat{x}$ be an approximate (within a factor of $1 + \varepsilon$) solution to this linear program. Later, we show how to obtain $\hat{x}$. In order to define a packing which is based on this solution, we define $y_C = \lceil \hat{x}_C \rceil$, for every configuration $C \in \mathcal{C}$. It can be seen that the vector $y$ is a feasible solution to the linear program (since $y_C \geq \hat{x}_C$ for all $C$, it satisfies all the constraints). Our scheme returns a solution that packs $y_C$ bins with configuration $C$, in this solution, there are at least $n(v)$ slots for every $v \in H$. Note that some of these slots may remain empty, which happens in the case that the number of slots is strictly larger than $n(v)$. As for the small items, they are sorted in non-increasing order according to size, and are packed using NEXT FIT (NF) into the bins that are packed using configurations in $\mathcal{C}_2$. In order to keep the packing feasible, NF moves on to the next bin just after a total size of 1 is reached or exceeded. The smallest item packed in each bin is the last item. Therefore, the removal of the smallest item brings the bin to a total size of less than 1. When this process is completed, the total space that small items occupy is at least $\sum_{C \in \mathcal{C}_2} (\Delta(C) - \varepsilon) y_C \geq \sum_{C \in \mathcal{C}_2} (\Delta(C) - \varepsilon) \hat{x}_C$, since these bins are occupied by a total size of at least 1 (and not exactly $1 + \varepsilon$). The remaining items are packed using NF into new bins. The total size of these items is at most $\varepsilon \sum_{C \in \mathcal{C}_2} y_C$, and every new bin would contain a total size of at least 1, therefore the cost of the solution

increases by a multiplicative factor of at most $1+\varepsilon$ as a result. One additional bin may result from this process since the last new bin (used for small items only) may be packed to a level smaller than 1.

To get a solution for $L \setminus L_1$, each item of the rounded-up instance is replaced by the corresponding item of $I$. We clearly use at most $(1+\varepsilon)\sum_{C\in\mathcal{C}} y_C + 1$ bins in this way. The items of $L_1$ are packed into dedicated bins, one item per bin. We later analyze the cost of this packing.

To solve the above linear program approximately, we invoke the column generation technique of Karmarkar and Karp [7]. We next elaborate on this technique. The above linear program has an exponential number of variables and a polynomial number of constraints (neglecting the non-negativity constraints). Instead of solving the linear program, we solve its dual program (that has a polynomial number of variables and an exponential number of constraints). The variables $z_v$ correspond to the item sizes in $H$, their intuitive meaning can be seen as weights of these items. An additional variable $Z$ corresponds to the small items.

$$\begin{aligned}
\max \quad & \sum_{v\in H} n(v)z_v + \Delta\cdot Z \\
s.t. \quad & \sum_{v\in H} n(v,C)z_v \le 1 && \forall C\in\mathcal{C}_1 && (3)\\
& \sum_{v\in H} n(v,C)z_v + \Delta(C)\cdot Z \le 1 && \forall C\in\mathcal{C}_2 && (4)\\
& z_v \ge 0 && \forall v\in H\\
& Z \ge 0.
\end{aligned}$$

To be able to apply the ellipsoid algorithm, in order to solve the above dual problem within a factor of $1+\varepsilon$, it suffices to show that there exists a polynomial time algorithm (polynomial in $n$ and $\frac{1}{\varepsilon}$) such that for a given solution $(\zeta^*) = (z^*, Z^*)$ (which is a vector of length $|H|+1 \le \frac{1}{\varepsilon^2}+1$, that is, $z^*$ is a vector of length $|H|$ and $Z^*$ is a scalar), decides whether $\zeta^*$ is a feasible dual solution (approximately). That is, it has one of two outputs. For the first possible output, it provides a configuration $C\in\mathcal{C}$ that has the following properties. If $C\in\mathcal{C}_1$, then $\sum_{v\in H} n(v,C)z_v^* > 1$, and if $C\in\mathcal{C}_2$, then $\sum_{v\in H} n(v,C)z_v^* + \Delta(C)Z^* > 1$. The second possible output is that an approximate infeasibility evidence does not exist, that is, for all configurations $C\in\mathcal{C}_1$, $\sum_{v\in H} n(v,C)z_v^* \le 1+\varepsilon$ holds, and for all configurations $C\in\mathcal{C}_2$, $\sum_{v\in H} n(v,C)z_v^* + \Delta(C)Z^* \le 1+\varepsilon$ holds. In this last case, $\frac{\zeta^*}{1+\varepsilon}$ is a feasible dual solution that can be used. If a configuration whose constraint in the dual linear program is violated, is found, we can continue with the application of the ellipsoid algorithm.

Such a configuration $C$ can be found using an approximated solution for the knapsack problem. The knapsack problem [11] is known to admit an FPTAS. In the standard knapsack problem, a set of items with sizes and weights are given, and the goal is to find a maximum weight subset of a total size at most 1. It is not difficult to adapt the solution so that the goal is to find a maximum weight subset with a total size strictly smaller than 1.

We employ this FPTAS for different inputs stated below. The goal is to distinguish between two cases. The first case is that there exists a configuration $C$ for which its constraint, which is either from the family (3) or from the family (4) is not satisfied. Otherwise, all these constraints are satisfied approximately, that is, the inequality is satisfied with $1+\varepsilon$ rather than with 1.

We first search for a configuration $C\in\mathcal{C}_1$ which maximizes $\sum_{v\in H} n(v,C)z_v^*$. $\mathcal{C}_1$ can be partitioned into subsets according to the size of the minimum item in it. For each $v\in H$, we search for a partial configuration $C'\in P(v)$ that maximizes $\sum_{u\in H} n(u,C')z_u^*$. The corresponding knapsack problem consists of the following input. The multi-set of items consists of the items of sizes $H(v) = \{u\in H | u\ge v\}$, where the number of items of size $u\in H(v)$ is $n(u)$. The weight of an item of size $u\in H(u)$ is $z_u^*$. The goal is to find a subset of items of total size less than 1 with maximum total weight. Let $M(v)$ be the weight of the output

of the FPTAS. The total weight of the items in the complete configuration (i.e., including an additional item of size $v$) is $M(v) + z_v^*$. If $M(v) + z_v^* > 1$, then there exists a complete configuration in $C(v)$ which does not satisfy the constraint.

We next search for a configuration $C \in \mathcal{C}_2$ which maximizes $\sum_{u \in H} n(u, C) z_u^* + \Delta(C) Z^*$, or

$$\sum_{u \in H} n(u, C) z_u^* + \left(1 + \varepsilon - \sum_{u \in H} n(u, C) u\right) Z^* = \sum_{u \in H} n(u, C)(z_u^* - uZ^*) + (1 + \varepsilon) Z^*.$$

Which is equivalent to maximizing $\sum_{u \in H} n(u, C)(z_u^* - uZ^*)$.

The knapsack problem here consists of all items of $H$, where $u \in H$ appears $n(u)$ times in the input, and the weight of such an item is $z_u^* - uZ^*$. The goal is to find a subset of items of total size less than 1 with maximum total weight. Let $M$ be the output of the FPTAS. That is, the total weight of the items in the complete configuration (i.e., including an additional item of size $v$) is $M$. If $M + (1 + \varepsilon) Z^* > 1$, then there exists a complete configuration in $\mathcal{C}_2$ which does not satisfy the constraint.

We show that if no such configuration was found, then $\frac{\zeta^*}{1+\varepsilon}$ satisfies all constraints. Since the FPTAS for the knapsack problem is a $1 + \varepsilon$ approximation, we can prove the following in the two cases. For a configuration $C \in C(v)$, we have $\sum_{u \in H} n(u, C) z_u^* - z_v^* \leq (1 + \varepsilon)(1 - z_v^*) \leq 1 + \varepsilon - z_v^*$ which implies $\sum_{u \in H} n(u, C) \frac{z_u^*}{1+\varepsilon} \leq 1$. For a configuration $C \in \mathcal{C}_2$, we have $\sum_{u \in H} n(u, C) z_u^* + \Delta(C) Z^* - (1 + \varepsilon) Z^* = \sum_{u \in H} n(u, C)(z_u^* - uZ^*) \leq (1 + \varepsilon)(1 - (1 + \varepsilon) Z^*) \leq 1 + \varepsilon - (1 + \varepsilon) Z^*$ which implies $\sum_{u \in H} n(u, C) \frac{z_u^*}{1+\varepsilon} + \Delta(C) \frac{Z^*}{1+\varepsilon} \leq 1$.

Since the approximated separation oracle that we described above runs in time which is polynomial in $n$ and $\frac{1}{\varepsilon}$, we conclude that the approximated solution of the (primal) linear program $\hat{x}$ is obtained in polynomial time (again polynomial in $n$ and $\frac{1}{\varepsilon}$). Since $\hat{x}$ is a solution of a linear program with an exponential number of variables, $\hat{x}$ is given in a compact representation, which is a list of non-zero components of the solution, together with their values.

To analyze the packing defined above, note that there are $|L_1| = \lceil |L| \varepsilon^2 \rceil \leq |L| \varepsilon^2 + 1$ bins which are used for the items of $L_1$. Each feasible bin may contain up to $\frac{1}{\varepsilon}$ large items. This is true since given a packed bin, the removal of one large item brings the total size of items below 1, and thus at most $\frac{1}{\varepsilon} - 1$ items are left after this removal. Therefore, $\text{OPT} \geq \varepsilon |L|$. We conclude that the number of additional bins (used to pack $L_1$) is at most $\varepsilon \text{OPT} + 1$, so the cost of the solution is at most $(1 + \varepsilon) \sum_{C \in \mathcal{C}} y_C + 1 + |L_1| \leq (1 + \varepsilon) \sum_{C \in \mathcal{C}} y_C + \varepsilon \text{OPT} + 2$. Therefore, it suffices to bound the cost, implied by $y$, in terms of OPT.

Instead of using an optimal solution to the linear program (whose value is a lower bound on $\text{OPT}(I')$, since $\text{OPT}(I')$ is a valid solution to the linear program), we use a $(1 + \varepsilon)$-approximated solution, and this degrades the value of the returned solution within a factor of $1 + \varepsilon$ (i.e., $\sum_{C \in \mathcal{C}} \hat{x}_C \leq (1 + \varepsilon) \cdot \text{OPT}(I')$).

We next bound $\sum_{C \in \mathcal{C}} (y_C - \hat{x}_C)$. Note that in the primal linear program there are at most $\frac{1}{\varepsilon^2} + 1$ constraints (not including non-negativity constraints), and hence in a basic solution (a property that we can always assume that $\hat{x}$ satisfies) there are at most $\frac{1}{\varepsilon^2} + 1$ positive components, and hence there are at most $\frac{1}{\varepsilon^2} + 1$ fractional components. Therefore, $\sum_{C \in \mathcal{C}} (y_C - \hat{x}_C) \leq \frac{1}{\varepsilon^2} + 1$.

Therefore, the cost of the obtained solution is at most $(1 + \varepsilon) \sum_{C \in \mathcal{C}} y_C + \varepsilon \text{OPT} + 2 = \sum_{C \in \mathcal{C}} (1 + \varepsilon) \hat{x}_C + \sum_{C \in \mathcal{C}} (1 + \varepsilon)(y_C - \hat{x}_C) + \varepsilon \text{OPT} + 2 \leq (1 + \varepsilon)^2 \cdot \text{OPT}(I') + (\frac{1}{\varepsilon^2} + 1)(1 + \varepsilon) + \varepsilon \text{OPT} + 2 \leq (1 + 4\varepsilon) \text{OPT} + \frac{6}{\varepsilon^2}$ (since $\varepsilon \leq 1$). Hence, we conclude the following theorem.

**Theorem 4** *The above scheme is an AFPTAS for* SOBP.

## 4 Conclusion

We studied several NP-hard open-end packing problems. We have shown that SOBP and LBC admit asymptotic fully polynomial approximation schemes, but the complexity of approximating OOBP remains open. Specifically, we have shown OOBP is strongly NP-hard, but it is unknown whether an AFPTAS (or even an APTAS) could be designed for it.

## References


[1] S.F. Assmann, D.S. Johnson, D.J. Kleitman, and J.Y.-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5:502–525, 1984.

[2] J. Csirik, D. S. Johnson, and C. Kenyon. Better approximation algorithms for bin covering. In *Proc.of the 12th Annual Symposium on Discrete Algorithms (SODA2001)*, pages 557–566, 2001.

[3] J. Csirik and V. Totik. On-line algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, 21:163–167, 1988.

[4] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[5] L. Gai and G. Zhang. It is hard to be lazy! manuscript, 2008.

[6] K. Jansen and R. Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1-3):543–551, 2003.

[7] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 312–320, 1982.

[8] J. Y.-T. Leung, M. Dror, and G. H. Young. A note on an open-end bin packing problem. *Journal of Scheduling*, 4(4):201–207, 2001.

[9] M. Lin, Y. Yang, and J. Xu. Improved approximation algorithms for maximum resource bin packing and lazy bin covering problems. In *Proc. of the 17th International Symposium on Algorithms and Computation (ISAAC2006)*, pages 567–577, 2006.

[10] M. Lin, Y. Yang, and J. Xu. On lazy bin covering and packing problems. In *Proc. of the 12th Annual International Conference on Computing and Combinatorics (COCOON2006)*, pages 340–349, 2006.

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

[12] J. Yang and J. Y-T. Leung. The ordered open-end bin packing problem. *Operations Research*, 51(5):759–770, 2003.

[13] G. Zhang. Parameterized on-line open-end bin packing. *Computing*, 60(3):267–274, 1998.