Bin packing with general cost structures

Leah Epstein*   Asaf Levin [†]

Abstract

Following the work of Anily et al., we consider a variant of bin packing, called BIN PACKING WITH GENERAL COST STRUCTURES (GCBP) and design an asymptotic fully polynomial time approximation scheme (AFPTAS) for this problem. In the classic bin packing problem, a set of one-dimensional items is to be assigned to subsets of total size at most 1, that is, to be packed into unit sized bins. However, in GCBP, the cost of a bin is not 1 as in classic bin packing, but it is a non-decreasing and concave function of the number of items packed in it, where the cost of an empty bin is zero. The construction of the AFPTAS requires novel techniques for dealing with small items, which are developed in this work. In addition, we develop a fast approximation algorithm which acts identically for all non-decreasing and concave functions, and has an asymptotic approximation ratio of 1.5 for all functions simultaneously.

1 Introduction

Classic bin packing [29, 11, 9, 10] is a well studied problem which has numerous applications. In the basic variant of this problem, we are given n items of size in $(0, 1]$ which need to be assigned to unit size bins. Each bin may contain items of total size at most 1, and the goal is to minimize the number of bins used.

Consider the following possible application. A multiprocessor system, where each bin represents one processor, is available for one unit of time. However, a processor that executes a large number of short tasks causes the system a larger load than a processor that executes a smaller number of long tasks, even if the total duration of the tasks is equal in both cases. This is one motivation to the problem BIN PACKING WITH GENERAL COST STRUCTURES (GCBP) that we study here. The problem has additional applications in reliability, quality control and cryptography [1].

In the problem GCBP, the cost of a bin is not 1, but it depends on the number of items actually packed into this bin. More precisely, we define the problem as follows. The input consists of n items $I = \{1, 2, \ldots, n\}$ with sizes $1 \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$, and a function $f : \{0, 1, 2, \ldots, n\} \to \mathbb{R}_0^+$, where f is a monotonically non-decreasing concave function, for which $f(0) = 0$. The goal is to partition I into some number of sets $S_1, \ldots, S_\mu$, called bins, such that $\sum_{j \in S_i} s_j \leq 1$ for any $1 \leq i \leq \mu$, and so that $\sum_{i=1}^{\mu} f(|S_i|)$ is minimized. We say that a function f is *valid* if it has the properties above, and an instance of GCBP is defined not only by its input item sizes but also using the function f. Without loss of generality we assume that $f(1) = 1$ (otherwise we can apply scaling to the cost function f).

---

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

[†]Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, Haifa, Israel. `levinas@ie.technion.ac.il`.

Anily, Bramel and Simchi-Levi [1] introduced GCBP and described a number of applications in detail. We describe their results in what follows. Further results on GCBP appear in [6], but these additional results are not related to this paper. A related model was studied by Li and Chen [25]. In this model the cost of a bin is a concave and monotonically non-decreasing function of the *total size* of items in it. Note that this variant is closely related to variable sized bin packing [26, 17].

We use the following notations for approximation algorithms. For an algorithm $\mathcal{A}$ for a given problem, we denote its cost by $\mathcal{A}$ as well. The cost of an optimal algorithm for the same problem is denoted by OPT. We define the asymptotic approximation ratio of an algorithm $\mathcal{A}$ as the infimum $\mathcal{R} \geq 1$ such that there exists a constant c, which is independent of the input, so that any input satisfies $\mathcal{A} \leq \mathcal{R} \cdot \text{OPT} + c$. The absolute approximation ratio of an algorithm $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \text{OPT}$. 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 $\frac{1}{\varepsilon}$. Polynomial time approximation schemes and fully polynomial time approximation schemes, which are abbreviated as PTAS and FPTAS, are defined similarly, but are required to give an approximation ratio of $1 + \varepsilon$ according to the absolute approximation ratio.

Anily, Bramel and Simchi-Levi [1] analyzed the worst case performance of some natural bin-packing heuristics when they are applied for GCBP (see Section 2 for the definitions of the heuristics). They showed that many common heuristics for bin packing, such as First Fit (FF), Best Fit (BF) and Next Fit (NF), do not have a finite asymptotic approximation ratio. Even an application of the first two heuristics on lists of items that are sorted by size in a non-increasing order, i.e., the algorithms First Fit Decreasing (FFD) and Best Fit Decreasing (BFD), leads to similar results. However, Next Fit Decreasing (NFD) behaves differently, and was shown to have an asymptotic approximation ratio of exactly 2. Sorting the items in the opposite order gives a better asymptotic approximation ratio of approximately 1.691 (in this case, the three algorithms First Fit Increasing (FFI), Best Fit Increasing (BFI) and Next Fit Increasing (NFI) are the same algorithm). Note that these heuristics are independent of the specific function f. It is stated in [1] that any heuristic that is independent of f has an asymptotic approximation ratio of at least $\frac{4}{3}$. Therefore, finding an algorithm with a smaller asymptotic approximation ratio, and specifically, an asymptotic approximation scheme, requires a strong usage of the specific function f.

In this paper, we develop an AFPTAS for GCBP. We develop a framework, where the action of the scheme for a given non-decreasing concave function f with $f(0) = 0$ is based on its exact definition. We also develop a new approximation algorithm MATCHHALF (MH), which acts obliviously of f, similarly to the behavior of the algorithms of [1]. We prove that our algorithm has an asymptotic approximation ratio of at most 1.5 for any non-decreasing concave function f with $f(0) = 0$, improving over the tight bound of approximately 1.691, proved by Anily et al. [1], on the asymptotic approximation ratio of NFI.

The classic bin packing problem is clearly a special case of GCBP as one can set $f(0) = 0$ and $f(i) = 1$ for all $i \geq 1$, where the resulting function is monotonically non-decreasing and concave. Therefore, GCBP inherits the hardness proof of the classic bin packing problem. That is, GCBP cannot be approximated within an absolute approximation ratio better than $\frac{3}{2}$ (unless $P = NP$). However, this does not exclude the possibility of the existence of asymptotic approximation schemes. We use the asymptotic approximation ratio as the main analytic tool to study approximation algorithms for GCBP, similarly to previous work for this problem [1] and other bin packing problems (see e.g.

[29, 19, 20, 21, 12, 22, 18, 3, 4]). In this metric we design the best possible result (assuming $P \neq NP$), i.e., an AFPTAS.

Fernandez de la Vega and Lueker [12] showed that the classic bin packing problem admits an APTAS. This seminal work introduced rounding methods which are suitable for bin packing problems. These methods, which were novel at that time, are widely used nowadays. Karmarkar and Karp [22] employed these methods together with column generation and designed an AFPTAS (see also [26, 28]). In [16], the complexity of two variants of bin packing with unit sized bins are resolved, that is, an AFPTAS is designed for each one of them. The first one is *Bin packing with cardinality constraints* [23, 7, 14], in which an additional constraint on the contents of a bin is introduced. Specifically, there is a parameter k which is an upper bound on the number of items that can be packed in one bin. The goal is as in classic bin packing, to minimize the number of bins used. The second one is *Bin packing with rejection* [15, 5, 13], in which each item has a rejection penalty associated with it (in addition to the size). Each item has to be either packed or rejected, and the goal is to minimize the sum of the following two factors: the number of bins used for the packed items and the total rejection penalty of all rejected items. Note that prior to the work of [16], these two problems were already known to admit an APTAS [7, 15, 5]. The main new tool, used in [16], which allows the design of schemes whose running time is polynomial in $\frac{1}{\varepsilon}$, is a treatment for small items using new methods developed in that work. The treatment of small enough items for the classic problem is rather simple. Roughly, the small items can be put aside while finding a good approximate solution, and can be added later in any reasonable fashion. Already in [7], it was shown that if the same treatment is applied to small items in the case of cardinality constraints, this leads to poor approximation ratios. Therefore, Caprara, Kellerer and Pferschy [7] developed an alternative method for dealing with small items. This method still separates the packing of large items from the packing of small items. The scheme enumerates a large number of potential packings of the large items, and for each packing, tests the quality of a solution that is constructed by adding the small items to the packing in a close to optimal way. The enumeration prevents this method from being used for designing algorithms with running time which is polynomial in $\frac{1}{\varepsilon}$. The way to overcome this difficulty, used in [16], is to find a good packing of large items, that takes into account the existence of small items, and allocates space for them. The packing of large items is typically determined by a linear program, therefore, the linear program needs to define at least some properties for the packing of small items. Specifically, the linear program does not decide on the exact packing of small items, but only on the type of a bin that they should join, where a type of a bin is defined according to the size of large items in the bin for bin packing with rejection, and on both the size and number of large items, for bin packing with cardinality constraints.

The problem studied in this paper, GCBP, is more complex than the ones of [16] in the sense that the cost of a bin is not just 1. Therefore, even though cardinality constraints are not present, the number of items packed into each bin must be controlled, in order to be able to keep track of the cost of this bin. In classic bin packing, and other well known variants, forcing all the bins of a solution to be completely occupied results in a perfect solution. To demonstrate the difficulty of GCBP, we show the existence of a non-decreasing concave function f with $f(0) = 0$, for which such a solution may still lead to a poor performance with respect to f.

In our scheme, cardinality constraints are implied by an advanced decision on the cost that needs to be paid for a given bin, that becomes a part of the type of the bin. The specific packing of small items, which is based on the output of the linear program, needs to be done carefully, so that the solution remains feasible, and to avoid large increases in the cost of the solution. An additional new ingredient used in our AFPTAS is a pre-processing step, which is performed on small items, where some of them are packed in separate bins which are not used for any other items. In typical packing

problems, bins which contain only very small items are relatively full, and thus the additional cost from such bins is close to the total size of these items. However, in our case, such a bin usually contains many items, and may result in a high cost. Therefore, our scheme always packs some portion of the smallest items separately, before any methods of packing items through a linear program are invoked. We show that the increase in the cost of the solution, due to the pre-processing step, is small enough, yet this allows more flexibility in the treatment of other small items, i.e., an additional bin would have a small cost compared to OPT.

The structure of the paper is as follows. In Section 2 we supply examples showing the unique nature of the problem GCBP, accompanied with new properties and some properties used in previous work. We use all these properties later in the paper. We introduce our fast approximation algorithm and analyze it in Section 3. Our main result is given in Section 4.

2 Preliminaries

In this section we demonstrate the differences between classic bin packing problems, and GCBP. We also state some properties proved in [1] and [2] to be used later.

As mentioned in the introduction, common heuristics do not have a finite approximation ratio for GCBP [1], and other heuristics have a higher approximation ratio than one would expect. Another difference is that sorting items in a non-decreasing order of their sizes is better than a non-increasing order.

We start with formal definitions of the heuristics. First Fit (FF), Best Fit (BF), and Next Fit (NF) act on arbitrarily ordered lists of items, processing the items and assigning them to bins in the order which they are given in the list. FF assigns each item into the bin of minimum index which can receive this item, and if no such bin exists, it opens a new bin and packs the item into the new bin. BF assigns each item into the bin which can receive this item and has the least empty space, and if no such bin exists, it opens a new bin and packs the item into the new bin. NF keeps a single active bin at each time. If an item can be packed into the active bin, then NF does that. Otherwise a new active bin is created and the item is packed there.

FFD, BFD, and NFD are the variants of FF, BF, and NF, respectively, which sort the items by non-increasing size and process them in this order. FFI, BFI, and NFI are the variants of FF, BF, and NF, respectively, which sort the items by non-decreasing size and process them in this order. As mentioned in the introduction, once the input is sorted by non-increasing size, the last three algorithms are in fact equivalent since an item which cannot be packed into a bin implies that any subsequent item (which cannot be smaller) cannot be packed in it.

A class of (concave and monotonically non-decreasing) functions $\{f_q\}_{q \in \mathbb{N}}$ that was considered in [1] is the following. These are functions that grow linearly (with a slope of 1) up to an integer point q, and are constant starting from that point. Specifically, $f_q(t) = t$ for $t \leq q$ and $f_q(t) = q$ for $t > q$. It was shown in [1] that focusing on such functions is sufficient when computing upper bounds on algorithms that act independently of the cost function. Note that $f_1 \equiv 1$, and thus GCBP with the cost function f_1 is equivalent to classic bin packing.

For an integer $K > 2$, consider inputs consisting of items of two sizes; $a = 1 - \frac{1}{K}$, and $b = \frac{1}{K^2}$.

Assume first that there is a single item of size a, and $2K$ items of size b. NFD packs the large item together with K of the small items in one bin, and additional K items in another bin. Consider the function f_K. The cost of the solution with respect to this function is $f_K(K + 1) + f_K(K) = 2K$. A solution that packs all small items in one bin and the large item in another bin has a cost of

$f_K(1) + f_K(2K) = K + 1$. Thus, even though both packings use the same number of bins, the cost of the first packing, which is produced by NFD, is larger by a factor that can be made arbitrarily close to 2, than the cost of the second packing. Moreover, even though only two bins are used, this proves a lower bound of 2 on the *asymptotic* approximation ratio of NFD (this bound is tight due to [1]).

Assume now that there are K items of size a and K^2 items of size b. An optimal packing for the classic bin packing problem clearly consists of K bins, such that each one is packed with one large item and K small items. Using the function f_K, this gives a cost of K^2. A different packing collects all small items in one bin, and has the cost $K \cdot f_K(1) + f_K(K^2) = 2K$. Since K can be chosen to be arbitrarily large, we get that the first packing, which is the unique optimal packing in terms of the classic bin packing problem, does not have a finite approximation ratio. Note that this first packing would be created by FFD and BFD, and also by FF, BF and NF, if the input is sorted appropriately.

Throughout the paper, if a specific cost function f is considered, we use OPT to denote the cost of an optimal solution OPT for the original input, which is denoted by I, with respect to f. Similarly, we use $\mathcal{A}(I)$ (or $\mathcal{A}$) to denote the cost of an algorithm $\mathcal{A}$ for the original input with respect to f. Recall that for an input J we use OPT(J) to denote both an optimal solution (with respect to f) for the input J (where J is typically an adapted input), and its cost. Thus OPT = OPT(I). We let $f_k(\mathcal{A}(I))$ be the cost of an algorithm $\mathcal{A}$ on I, calculated with respect to the cost function f_k, and abbreviate it by $f_k(\mathcal{A})$. If $f = f_k$ then we have $\mathcal{A} = f_k(\mathcal{A})$ and OPT $= f_k(\text{OPT})$. Thus $f_1(\mathcal{A}(I))$ (or $f_1(\mathcal{A})$) is the number of bins used by $\mathcal{A}$ on the input I, but note that $f_1(\mathcal{A})$ is equal to $\mathcal{A}$ only if $f = f_1$, that is, if the classic bin packing problem is considered. The minimum number of bins required to pack I (i.e., the cost of an optimal solution for the classic bin packing with the input I) is denoted by $f_1(\text{OPT}(I))$ or $f_1(\text{OPT})$.

For the analysis of our fast heuristic, we use weighting functions. This type of analysis was widely used for classic bin packing, and many variants of bin packing. The basic technique was used as early as in 1971 by Ullman [29] (see also [21, 24, 27]). Specifically, the following theorem will be used.

Theorem 1 *Consider an algorithm $\mathcal{A}$ for classic bin packing. Let w_1, w_2 be two weight measures defined on the input items, $w_i : I \to \mathbb{R}$, for $i = 1, 2$. Let $W_1(I)$ and $W_2(I)$ denote the sum of weights of all input items of I, according to w_1 and w_2 respectively, and assume $W_2(I) \leq W_1(I)$. Assume that for every input of the algorithm, the number of bins used by the algorithm $\mathcal{A}$ is at most $W_2(I) + \tau$, for a constant value τ which is independent of I. Denote by W_I the supremum total weight of a set of items that can be packed into a single bin of the optimal solution, according to measure w_1. Then the asymptotic approximation ratio of $\mathcal{A}$ is no larger than W_I.*

Proof. Given an input I we have $\mathcal{A} \leq W_2(I) + \tau$. Since an optimal algorithm has OPT(I) bins, with a weight of at most W_I in each one of them, we get the upper bound on the weight, according to w_1; $W_1(I) \leq W_I \cdot \text{OPT}(I)$. Using $W_2(I) \leq W_1(I)$, we get $\mathcal{A} \leq W_I\text{OPT}(I) + \tau$ and the theorem follows. ∎

We will make use of adaptations of the following function $w : [0, 1] \to \mathbb{R}$ (that is equal to the function $W_1(p)$ defined in [2] for any $p > 0$). We first define the well known sequence π_i, $i \geq 1$, which often occurs in bin packing.

Let $\pi_1 = 2$, and for $i \geq 1$,

$$\pi_{i+1} = \pi_i(\pi_i - 1) + 1.$$

Thus $\pi_2 = 3$, $\pi_3 = 7$, $\pi_4 = 43$, etc. For $p \in (\frac{1}{k+1}, \frac{1}{k}]$, we define

$$w(p) = \frac{1}{k},$$

if $k = \pi_i - 1$ for some $i \geq 1$, and otherwise,

$$w(p) = \frac{k+1}{k} \cdot p.$$

Finally, we let

$$w(0) = 0.$$

Note that w is a monotonically non-decreasing function. It was shown in [2] that for a given input I, $\sum_{i \in I} w(s_i) \geq f_1(\text{NFD}(I)) - 3$. Even though both [2] and [1] assume that no zero sized items exist, clearly, the number of bins used by NFD and NFI does not increase as a result of the existence of such items, unless all input items are of size zero, and therefore, this property on the weights still holds even if zero sized items are allowed.

We further state some lemmas proved in [1] that allow us to simplify our analysis in the next section.

Lemma 2 *[Property 3 in [1]]* $f_1(\text{NFI}(I)) = f_1(\text{NFD}(I))$, *and therefore* $\sum_{i \in I} w(s_i) \geq f_1(\text{NFI}(I)) - 3$.

Lemma 3 *[Theorem 1 in [1]] Consider a packing heuristic $\mathcal{A}$ that does not use information on the function f. If the asymptotic approximation ratio of $\mathcal{A}$ is at most $\mathcal{R}$, for any function f_k (for $k \geq 1$), then the asymptotic approximation ratio of $\mathcal{A}$ is at most $\mathcal{R}$ for any non-decreasing concave function f with $f(0) = 0$.*

A useful packing concept, defined in [1], is *consecutive bins*. Recall that we assume $s_1 \geq s_2 \geq \cdots \geq s_n$. Let $B_1, B_2, \ldots, B_\mu$ be the subsets of items packed into the bins created in some solution $\mathcal{B}$ that packs the items in μ bins, where B_i is the i-th bin. The packing has consecutive bins if the union $\cup_{j \leq s} B_j$ is a suffix of the sequence $1, 2, \ldots, n$ for any $1 \leq s \leq \mu$. That is, if the first s bins contain n' items, then these are the n' items $n - n' + 1, \ldots, n - 1, n$ (and thus the smallest n' items). The following lemma states that NFI is the *best* heuristic among such with consecutive bins. Consider a given input I, the cost function f_k and a feasible packing with consecutive bins $\mathcal{B}$.

Lemma 4 *[Corollary 3 in [1]]* $f_k(\text{NFI}(I)) \leq f_k(\mathcal{B}(I))$.

A partition of the items (which is not necessarily a valid packing) with consecutive bins is called *an overflowed packing* if for all $1 < i < \mu$, $\sum_{j \in B_i} s_j > 1$. Clearly, if $\mu > 2$, such a packing must be infeasible. The following lemma implies a lower bound on the cost of an optimal solution. Consider a given input I, a cost function f_k, an overflowed packing with consecutive bins $\mathcal{B}$, and a feasible packing $\mathcal{A}$.

Lemma 5 *[Corollary 1 in [1]]* $f_k(\mathcal{B}(I)) \leq f_k(\mathcal{A}(I))$.

Using these properties, in order to analyze NFI, it is enough to consider the functions f_k for $k \geq 1$. It was shown in [1] that the asymptotic approximation ratio of NFI for the function f_k ($k \geq 2$) is at most $1 + \frac{1}{k}$. The asymptotic approximation ratio of NFI for f_1, that is, for classic bin packing, follows from the results of [2] and from Lemma 2. This ratio is $\sum_{i=1}^{\infty} \frac{1}{\pi_i - 1} \approx 1.691$. Thus the upper bound of 1.691 [1] follows. In the next section we use these properties to develop a new algorithm. The algorithm needs to carefully keep the approximation ratio for $k = 2$ while improving the approximation ratio for $k = 1$.

3 A fast approximation algorithm MH

In this section we describe a simple and fast algorithm MH, that does not need to know the function f in advance. This algorithm is a modification of NFI that tries to combine a part of the relatively large items (of size larger than $\frac{1}{2}$) in bins together with one additional item. Note that except for possibly one item, NFI packs all such items in dedicated bins.

As mentioned above, NFI has an asymptotic approximation ratio of at most $\frac{k+1}{k}$ for the function f_k with $k \geq 2$. Therefore, the difficult case is actually the classic problem. On the other hand, using heuristics that perform well for the classic problem, such as FFD, may lead to worse results for $k \geq 2$ (which in fact is the case for FFD). Therefore, we define an algorithm that acts identically to NFI, except for the usage of a pre-processing step.

Algorithm MATCHHALF (**MH**)

1. Let t be the number of items in I with size in $(\frac{1}{2}, 1]$ (which are called *large items*).

2. Let $M_0 = \{\lceil \frac{t+1}{2} \rceil, \ldots, t\}$, that is, M_0 is the set of smallest $\lceil \frac{t}{2} \rceil$ large items, and let $M_1 = \{1, \ldots, \lceil \frac{t-1}{2} \rceil\}$ be the remaining large items. Let $S = \{t+1, \ldots, n\}$ be called the set of small items.

3. Define the following bipartite graph. One set of vertices consists of the large items of M_0. The other set of vertices consists of all small items. An edge (a, b) between vertices of items of sizes $s_a > \frac{1}{2}$ and $s_b \leq \frac{1}{2}$ exists if $s_a + s_b \leq 1$, i.e., if these two items can be placed in a bin together. If this edge occurs, its cost is defined as $c(a, b) = w(b)$ (using the function w of Section 2).

4. Find a maximum cost matching in the bipartite graph. This matching can actually be found using the following greedy process. Insert the items of S into a queue in a sorted order, with item $t+1$ at the top, and the items M_0 are inserted into a queue in a sorted order with item t at the top. At each time, let j be the item at the top of the first queue, and i the item at the top of the second queue. If $s_i + s_j \leq 1$, these items are matched, and removed from the queues. Otherwise, item j cannot be matched to any item of the second queue (since s_i is minimal in that queue), so j is removed from the first queue. This process is done until one of the queues is empty, and is performed in linear time.

5. Each pair of matched items is removed from I. Every matched pair is packed into a bin together.

6. Pack the remaining items using NFI.

The greedy process of step 4 finds an optimal matching by a simple exchange argument. We note that only (approximately) half of the large items are possibly matched in the pre-processing step. A larger fraction may cause an asymptotic approximation ratio above 1.5, as can be seen in the following example. Let K be an integer such that $K > 2$. The input set I consists of K items of size $\frac{1}{K}$ and K items of size $1 - \frac{1}{K}$. Running NFI on this input results in one bin containing K items of size $\frac{1}{K}$ and K bins containing one larger item. However, if we match an α fraction (for some $0 \leq \alpha \leq 1$) of the larger items in a pre-processing step, there would be αK bins with two items. Consider the function f_2. We get $f_2(\text{NFI}(I)) = K + 2$, whereas the cost with pre-processing is at least $\alpha K + K$. This would give an approximation ratio of at least $1 + \alpha$.

We first analyze the asymptotic approximation ratio for f_1, and later we analyze it for all other functions f_k.

An analysis for f_1. In this case we use weighting functions for the analysis.

Lemma 6 *For any input I, $f_1(\text{MH}(I)) \le \frac{3}{2} f_1(\text{OPT}((I)) + 3$.*

Proof. To use Theorem 1, we define a weight measure w_2 on items as follows. For every item i, we let $w_2(i) = w(s_i)$, except for small items that are matched to large items in the pre-processing step of MH. These items receive a weight of zero according to w_2. Let X be the number of bins created by the pre-processing step and Y the number of bins created by NFI (i.e., in Step 6 of the algorithm). Let I' be the input after the removal of items in the pre-processing step. By Lemma 2, we have $\sum\limits_{i \in I'} w_2(i) = \sum\limits_{i \in I'} w(s_i) \ge Y - 3$. On the other hand, every bin created in the pre-processing step has a total weight of 1, since each such bin contains a large item (that has a weight of 1) and a small item of weight 0. Thus $\sum\limits_{i \notin I'} w_2(i) = X$ and in total $\sum\limits_{i \in I} w_2(i) \ge X + Y - 3 = f_1(\text{MH}) - 3$.

Next, we define a weight measure w_1. Consider the t large items, and their packing in an optimal solution OPT. For any large item a, which is packed in a bin with at least one other (small) item, consider the largest small item which is packed with a and denote it by z_a, breaking ties arbitrarily. If no such item exists, i.e., a is packed as a single item in a bin of OPT, we add an item of size zero to this bin of OPT and define it to be z_a. Therefore z_a exists and is defined uniquely for every large item a. We define the weight of every item i as

$$w_1(i) = w(s_i),$$

except for the items z_a for $a = 1, \ldots, t$, for which we let

$$w_1(z_a) = \frac{w(s_{z_a})}{2}.$$

In order to show $W_2(I) \le W_1(I)$, we define a valid matching in the auxiliary graph. This matching is based on the packing of OPT. Let $Z = \{z_a | 1 \le a \le t\}$ and denote a set of the largest $\lceil \frac{t}{2} \rceil$ items in $Z = \{z_a | 1 \le a \le t\}$ by Z'. We initialize the matching with the items of Z' being matched to the large items from their bins in OPT. This matching is valid since by definition of Z, each item in this set is packed in OPT in a different bin, with a different large item. If the $\lceil \frac{t}{2} \rceil$ items matched to them are not exactly items $\lceil \frac{t+1}{2} \rceil, \ldots, t$, it is possible to replace some large items in the matching by smaller large items, until this situation is reached. We have $s_{i_1} \le s_{i_2}$ for $i_1 \in Z \setminus Z'$ and $i_2 \in Z'$. Since the function w is monotonically non-decreasing, we get $\sum\limits_{z_a \in Z} w(s_{z_a}) \le 2 \sum\limits_{z_a \in Z'} w(s_{z_a})$. Let $W(I) = \sum\limits_{i=1}^{n} w(s_i)$. We have $W_2(I) = W(I) - c(M)$, where $c(M)$ is the cost of a matching in the auxiliary graph, with a maximum cost, and $W_1(I) = W(I) - \sum\limits_{1 \le a \le t} \frac{w(s_{z_a})}{2} \ge W(I) - \sum\limits_{z_a \in Z'} w(s_{z_a}) \ge W(I) - c(M) = W_2(I)$, since $c(M)$ is a maximum cost matching on the smallest $\lceil \frac{t}{2} \rceil$ large items, and $\sum\limits_{\lceil \frac{t+1}{2} \rceil \le a \le t} w(s_{z_a})$ is the cost of one such matching, which we defined above.

Finally, we need to find an upper bound on the total weight in a bin of OPT, according to w_1. We first consider bins that do not contain a large item. For any item i of size $s_i = \beta \in (0, \frac{1}{2}]$, we have

$w_1(i) \leq \frac{3}{2}\beta$. For items of size 0 the weight is 0. Therefore, the total weight of items in such a bin is no larger than 1.5 (a tighter upper bound of 1.423 is proved in [2]).

Consider next a bin which contains a large item. Let a be the large item of this bin, and z_a is chosen as above. If $s_{z_a} = 0$, then the only item in the bin that has a non-zero weight according to w_1 is a, and thus the total weight is 1. Otherwise, let j be such that $s_{z_a} \in (\frac{1}{j+1}, \frac{1}{j}]$. Any other item i in the bin (except for a and z_a) satisfies $w_1(i) \leq \frac{j+1}{j}s_i$ (since $s_i \leq s_{z_a} \leq \frac{1}{j}$). If $j = \pi_i - 1$ for some $i \geq 1$, we have $w_1(z_a) = \frac{1}{2j}$. Otherwise, $w_1(z_a) = \frac{j+1}{2j}s_{z_a}$.

We have a total weight of at most $1 + w_1(z_a) + \frac{j+1}{j}(1 - s_a - s_{z_a}) \leq 1 + w_1(z_a) + \frac{j+1}{j}(\frac{1}{2} - s_{z_a})$, since $s_a > \frac{1}{2}$. In the first case we use $s_{z_a} > \frac{1}{j+1}$, and get at most $1 + \frac{1}{2j} + \frac{j+1}{2j} - \frac{1}{j} = \frac{3}{2}$. In the second case we get at most $1 + \frac{j+1}{2j}s_{z_a} + \frac{j+1}{2j} - \frac{j+1}{j}s_{z_a} = \frac{3j+1}{2j} - \frac{j+1}{2j}s_{z_a}$. Using the same property we get at most $\frac{3}{2}$ again. ∎

An analysis for functions f_k with $k \geq 2$. Note that for a function f_k with $k \geq 2$ it holds that a bin packed with two items incurs exactly the same cost as two bins packed with a single item each. Let I be the original input on which MH is executed. Let $\hat{I}$ denote an input in which every small item, which is matched with a large item in the pre-processing step of MH, is replaced with an item of size s_1. Thus, at most $\lceil \frac{t}{2} \rceil$ items are increased to the size s_1. If $t = 0$, then $\hat{I} = I$, and otherwise $s_1 > \frac{1}{2}$. We consider the following solutions and compare their costs. Recall that the cost of the solution of MH on I, with respect to f_k, is denoted by $f_k(\text{MH}(I))$ and the cost of the solution of NFI on $\hat{I}$, with respect to f_k, is denoted by $f_k(\text{NFI}(\hat{I}))$. The next solution that we consider is an overflowed solution that is created for I as follows. The items are sorted by size in a non-decreasing order (that is, order by indices in a decreasing order). At each time, a minimum prefix of the items of total size larger than 1 is assigned to the next bin. We denote this solution by O and thus the cost of this solution with respect to f_k is denoted by $f_k(O(I))$. The cost of an optimal solution for I, with respect to f_k, is denoted by $\text{OPT}_k(I)$. Finally, we consider a solution for $\hat{I}$ with consecutive bins, which is constructed from the overflowed solution for I as follows (the construction is similar to the one in [1], except for the treatment of items in $\hat{I}$, and the fact that the corresponding items in I are simply removed). For every bin of the overflowed solution, if the total size of items exceeds 1 (this is the case with all bins except for possibly the last bin, or bins with removed items), remove the last item and open a new bin for it. The additional large items of $\hat{I}$, which existed as smaller items in I and were removed from I, are assigned to dedicated bins. We denote this solution by C and so the cost of this solution, with respect to f_k, is denoted by $f_k(C(\hat{I}))$.

By Lemma 4, we have $f_k(\text{NFI}(\hat{I})) \leq f_k(C(\hat{I}))$. By Lemma 5, we have $f_k(O(I)) \leq \text{OPT}_k(I)$. We next prove two lemmas after which we will be able to conclude $f_k(\text{MH}(I)) \leq \frac{3}{2}\text{OPT}_k(I) + 3.5$.

Lemma 7 $f_k(\text{MH}(I)) \leq f_k(\text{NFI}(\hat{I})) + 1$.

Proof. Since all small items of I that are packed in the pre-processing step of MH are large in $\hat{I}$, the small items packed by NFI in the two algorithms are the same ones, and bins created by NFI in the two algorithms are identical, except for bins that contain a large item. If any of the two applications of NFI outputs a bin that contains a large item together with other items, we adapt the solution by moving this item into a separate bin. This modification cannot decrease the cost of a solution, but it may increase the cost by at most 1. The small items bins, resulting from running NFI in both solutions (the solution of MH and the solution of NFI, possibly with the modification) are now identical. The remaining items

are packed in both solutions either in singles or in pairs. Thus the costs of such bins are equal in both solutions (since $k \geq 2$). Therefore, the claim follows. ∎

Lemma 8 $f_k(C(\hat{I})) \leq \frac{3}{2} f_k(O(I)) + 2.5$.

Proof. We first modify both solutions so that none of them combines large items with some small item in one bin (but the overflowed solution may still have bins with two large items, which are not modified here). For the overflowed solution, this may require moving one or two large items from a shared bin to a dedicated bin, so it may increase the cost by at most 2. For the other solution, this may involve moving one large item to a dedicated bin, and cannot decrease the cost of the solution. We consider first the bins with small items, that contain at least $k + 1$ items in the overflowed solution. For every such bin, its cost is at least k. As a result of moving the last item to a dedicated bin (in the process of creation of the feasible solution), an additional cost of at most 1 is incurred. Thus the cost increases by at most a factor of $\frac{3}{2}$. For any bin containing at most k items, there is no additional cost from this step. Note that all bins with large items are in this situation. The cost of bins with large items in the overflowed solution with the modification is simply t, no matter how they are exactly packed, so packing each one in a dedicated bin does not change the cost. Together with the additional $\lceil \frac{t}{2} \rceil$ large items, the cost of large items becomes $\lceil \frac{3t}{2} \rceil \leq \frac{3t}{2} + \frac{1}{2}$. Removing small items of I, whose size is different in $\hat{I}$, and therefore these items do not need to be packed in the solution C (they are already packed as items of size s_1 in dedicated bins), may only decrease the cost. This proves the claim. ∎

Using Lemma 3, we have proved the following.

Theorem 9 *The asymptotic approximation ratio of* MH *is at most 1.5 for any non-decreasing concave function f with $f(0) = 0$.*

We have shown above that for $k = 2$ (and similarly, for any constant k), the bound 1.5 is tight. Note that the bound 1.5 is tight for $k = 1$ as well. Consider an input with N large items of size $\frac{1}{2} + \frac{1}{2K}$, and $N(K-1)$ small items of size $\frac{1}{2K}$ (for large enough N, K, such that N is divisible by $4K$). MH creates $\frac{N}{2}$ bins with one large and one small item, $\frac{N(K-1) - \frac{N}{2}}{2K} = \frac{N(K - \frac{3}{2})}{2K}$ bins with $2K$ small items each, and $\frac{N}{2}$ bins with one large item. This gives a total cost of $N + \frac{N(K - \frac{3}{2})}{2K}$. An optimal solution combines $K - 1$ small items with every large item, for a cost of N. For large enough K, the ratio is arbitrarily close to 1.5. It can be seen that this ratio is achieved for any fraction $0 \leq \alpha \leq 1$ of large items that participate in the pre-processing step.

4 An AFPTAS for GCBP

In this section we present our main result, that is, an AFPTAS for GCBP. We give a sketch which presents the main ideas and technical difficulties, and give the full description of the AFPTAS and its analysis later. We first present an auxiliary algorithm called *Fractional Next-Fit Increasing*.

4.1 FNFI and its analysis

In this section we prove a property regarding the cost of optimal packings, which is helpful in the design of our AFPTAS. It is related to the property on NFI stated in Lemma 4, but it is stronger since it is proven for any non-decreasing concave function f with $f(0) = 0$, for fractional packing of items. A

packing is fractional if items can be cut into pieces, where pieces of one item can possibly be packed in different bins.

To be able to analyze fractional packings, we next define f for any (real and not necessarily integral) non-negative value.

Definition 1 *If $x \geq n$ then let $f(x) = f(n)$. Otherwise, we define $f(q)$ for $q \in [0, n]$ as follows. The values of f for integer values of q are unchanged. If $i < q < i + 1$, then*

$$f(q) = (i + 1 - q) \cdot f(i) + (q - i) \cdot f(i + 1).$$

This function is piecewise linear and continuous, and since it is an extension of a non-decreasing concave function on integers, it is monotonically non-decreasing and concave in $[0, n]$.

The cost of a fractional packing is calculated according to the *generalized* function f, while for each bin, f is applied on the *number of items* in this bin. The number of items in a bin which is packed fractionally is the sum of fractions in it. This number is not necessarily an integer and it is not related to sizes of these fractional items, but only to their fractions. More accurately, we define fractions as follows.

Definition 2 *If an item is packed in a bin completely, we say that its fraction packed in the bin is 1. If a part of size α of an item of size $\beta > 0$ is packed in a given bin, we say that the fraction of this item that is packed in this bin is $\frac{\alpha}{\beta}$.*

There is no advantage in packing fractions of size zero of items (which are not zero sized). As for zero sized items, it is possible to split such an item among several bins, but since f is concave, it is never profitable to do so. We assume without loss of generality that in every fractional packing, every bin contains at most one part of each item. If this property does not hold, it is possible to unite parts of items within a bin without changing the cost.

We consider an algorithm which creates a fractional packing of the items according to the variant of the NFI heuristic, called FRACTIONAL NFI (FNFI). This algorithm sorts items by size in non-decreasing order. At each time, a bin is filled completely, before moving on to the next bin. For this, we allow the splitting of items into several parts, that is, the last item that is packed in a bin is possibly just a part of an item. Consequently, the first item packed in the next bin may be the remaining part of the same item. Note that each bin in the output of FNFI contains at most two split items and that in total only at most $\mu - 1$ items are split (where μ is the number of bins used by FNFI). Note that FNFI packs all zero sized items into the very first bin.

A simple property of FNFI is that it creates bins that are sorted in a non-increasing order of the number of items in them. This holds since given two bins $i_1 < i_2$, bin i_1 is completely occupied, and every item that has a part packed in bin i_1 has a size no larger than any item that has a part packed in bin i_2.

For any non-decreasing concave function f with $f(0) = 0$, the following lemma states that FNFI is the best heuristic among packings with fractionally packed bins. Consider a given input I, a cost function f and a fractional packing, $\mathcal{B}$.

Lemma 10 $f(\text{FNFI}(I)) \leq f(\mathcal{B}(I))$.

Proof. Assume by contradiction that for an input I, a fractional packing $\mathcal{B}$ and a function f, we have $f(\text{FNFI}(I)) > f(\mathcal{B}(I))$. Assume that the bins of $\mathcal{B}$ are sorted according to a non-increasing numbers of items. If the packing $\mathcal{B}$ that satisfies the condition is not unique, consider such a packing $\mathcal{B}$ which

maximizes the suffix of bins that are packed identically to the packing of FNFI. Consider the first bin i of $\mathcal{B}$ that is packed differently from the packing of FNFI. If bin i is the very last bin of the packing $\mathcal{B}$, then the bins $1, \ldots, i-1$ are packed as in the packing of FNFI, and therefore, bin i also has the same contents for $\mathcal{B}$ as it has for FNFI. Therefore we assume that i is not the last bin of $\mathcal{B}$.

Let $j, j+1, \ldots, j'$ be the indices of items that FNFI packs in bin i (the first and last items, which have the indices j' and j respectively, may be packed fractionally in this bin). Let $j \leq j_1 \leq j'$ be an index of an item such that $\mathcal{B}$ packs a smaller part of j_1 (possibly of size zero) in bin i than FNFI does. Such an item must exist by the following argument. If FNFI fills bin i completely, then since bin i of $\mathcal{B}$ is packed differently, it cannot have at least the same fraction of every item. Otherwise, FNFI packs all the remaining items in bin i, so a different packing of bin i means that some item has a smaller fraction in $\mathcal{B}$.

We next consider the case that there exists an item j_2 for which $\mathcal{B}$ packs a larger part in bin i than the packing of FNFI. Since the two algorithms pack bins $1, \ldots, i-1$ identically, only the items of index up to j' are available for packing in bins $i, i+1, \ldots$, where the item of index j' may already be fractional. Out of these items, FNFI packs a maximum prefix into bin i, so this item must satisfy $j_2 \leq j$. We get that $j_2 \leq j \leq j_1$. Since $j_1 \neq j_2$ by their definitions, we get $j_2 < j_1$.

Denote the fractions of j_1 and j_2 in bin i of $\mathcal{B}$ by γ_1 and γ_2, and the fractions of j_1 and j_2 in bin i of FNFI by δ_1 and δ_2. We have $\delta_1 > \gamma_1 \geq 0$ and $\gamma_2 > \delta_2 \geq 0$. Since $\gamma_1 < \delta_1$, and bins $1, \ldots, i-1$ are packed identically in both algorithms, there exists a further bin i' that contains a part of item j_1 in the packing of $\mathcal{B}$. Let $\varepsilon_1 > 0$ be the fraction of j_1 in bin i' of $\mathcal{B}$.

We would like to swap parts of items in the packing of $\mathcal{B}$, specifically, a part of item j_1 from bin i' with a part of item j_2 in bin i. We use μ to denote the size of the swapped part. There are three restrictions on μ. The resulting fraction of j_1 in bin i of $\mathcal{B}$ cannot exceed the fraction of this item in bin i of FNFI, thus $\mu \leq (\delta_1 - \gamma_1)s_{j_1}$. We can swap at most a fraction ε_1 of j_1. Moreover, we can swap at most a fraction of $\gamma_2 - \delta_2$ of j_2, in order to keep a fraction of j_2 in bin i that is at least as large as the one in bin i of FNFI. Therefore, we let $\mu = \min\{(\gamma_1 - \delta_1)s_{j_1}, (\gamma_2 - \delta_2)s_{j_2}, \varepsilon_1 s_{j_1}\}$. We adapt $\mathcal{B}$ by swapping a part of size μ of item j_1 from bin i' with a part of size μ from j_2 in bin i. By definition of all variables, $\mu > 0$, and thus some change occurred.

Let n_i and $n_{i'}$ be the original numbers of items in bins i and i' of $\mathcal{B}$. By our assumption $n_i \geq n_{i'}$. Let α_1 and α_2 be the fractions of items j_1 and j_2 that are swapped. Since $\mu = \alpha_1 \cdot s_{j_1} = \alpha_2 \cdot s_{j_2}$, and $s_{j_1} \leq s_{j_2}$, we have $\alpha_1 \geq \alpha_2$. Thus, the change in the cost is $f(n_i - \alpha_2 + \alpha_1) + f(n_{i'} - \alpha_1 + \alpha_2) - f(n_i) - f(n_{i'}) \leq 0$, by concavity. As a result of this process, the total number of items in bin i remains no smaller than the numbers of items in each of the bins $i+1, i+2, \ldots$.

If an item j_2 does not exist, it means that bin i has a total size of items that is smaller than the total size of items in bin i of FNFI. In particular, it means that bin i is not fully packed. We define $\gamma_1, \delta_1,$ i' and ε_1 as before. In this case we can define $\mu = \min\{(\gamma_1 - \delta_1)s_{j_1}, \varepsilon_1 s_{j_1}\}$. We define α_1, n_i and $n_{i'}$ as before. Thus, the change in the cost is $f(n_i + \alpha_1) + f(n_{i'} - \alpha_1) - f(n_i) - f(n_{i'}) \leq 0$, by concavity.

It is possible to perform this process on bin i multiple times, until there is no item that has an item for which a smaller fraction of it is packed in bin i of $\mathcal{B}$ than it is packed in the same bin for FNFI. At this time these bins become identically packed.

We next show that this situation, where no item j_1 exists, is reached after a finite number of swaps. For every item j_1, it can be performed for every item j_2 and for every successive bin. This gives a total of at most n^3 swaps, and possibly n^2 movements of items to bin i without swaps.

After we reach the situation where bin i is identical for $\mathcal{B}$ and FNFI, the bins $1, \ldots, i$ of $\mathcal{B}$ are sorted by a non-increasing number of items. Each remaining bin of $\mathcal{B}$ has a number of items that is

no larger than bin i. Moreover, bins $i + 1, i + 2, \ldots$ can be sorted so that the list of bins becomes sorted as required. The changes above can only decrease the cost of the solution, and therefore we get a contradiction to our assumption. ∎

4.2   The sketch of the scheme

We define an item to be a small item if its size is smaller than ε and otherwise it is a large item. Denote the set of large items in I by L. The small items will be partitioned further into two sets S and M, where M is the set of medium items and S is the set of smallest items. Our first step is to apply linear grouping [12] of the large items, that is, we sort them by size and we partition them into $\frac{1}{\varepsilon^3}$ (almost) equal-sized sets of consecutive items (in the sorted list). We pack each item of the set of the largest items in its own bin, and we round up the size of the items in each other set to the largest size of an item in that set.

We next partition the items in $I \setminus L$ into $M \cup S$ where S contains the smallest items such that the total size of the items in S is close to a constant which we define depending on ε. The items of S are packed nearly optimally using the FNFI heuristic, packing any split item using a dedicated bin. These bins will enable us to use a constant number of bins with an arbitrary content (of items in $L \cup M$) while paying at most ε times the cost of the bins which are used to pack the items in S. We note that packing S using the NFI heuristic is also possible and leads to a similar performance guarantee. However, the analysis of using FNFI is simpler.

Our next step is to approximate the cost function f from below using a staircase (step) function with $O(\log f(n))$ steps. We use monotonicity and concavity of f to show that this number of steps in the function is sufficient to get a $(1 + \varepsilon)$-approximation of f. We note that $\text{OPT}(I) \geq f(n)$ holds for any concave function f.

We next move on to finding a packing of the items in $L \cup M$ (neglecting the largest items which are packed in dedicated bins). In such an instance, the linear program which we construct allows the items of M to be packed fractionally. To construct this linear program we define a set of configurations of large items (which is a relatively standard definition), and a set of extended configurations which also define the space and cardinality of small items in a configuration (which is a non-standard idea that was introduced in [16]). The linear program will decide how many bins with a given extended configuration to open and what type of bins each small item needs to be packed in. These types are called windows, and we define them as the pair consisting of the total space for the small items and the total cardinality of small items in a bin with this window. Hence in this linear program we have a constraint for each size of large items (a constant number of constraints), a constraint for each small item (a linear number of such constraints), and two constraints for each type of window. We apply the column generation technique of Karmarkar and Karp [22] to solve the resulting linear program approximately (we use a separation oracle which applies an FPTAS for the Knapsack problem with cardinality constraint given by [8]).

Unfortunately the number of fractional entries in a basic solution for this linear program (as we can assume our solution is indeed a basic solution) is linear in the number of windows types (plus a constant). The number of windows is indeed polynomial in the input size allowing us to solve the linear program, but it is not a constant, and we will incur a too large error if we would like to round up the fractional solution.

Hence, we define a restricted set of windows types with a much smaller set of windows, and we show how to carefully project our solution to a new solution which is not worse than the original solution, and whose support uses only windows from this restricted set of windows. Therefore, when

we count the number of constraints, we can eliminate the constraints corresponding to windows which do not belong to the restricted set of windows. Thus the new bound on the number of fractional components in the projected solution is now much smaller. That is, our projected solution which is an approximated solution to the restricted linear program gives also an approximated solution to the original linear program with additional constraints by setting the variable to zero if the corresponding window does not belong to the restricted set of windows.

The next step is to round up the resulting projected solution. If a small item is packed fractionally, then we pack it in its own dedicated bin. If the fractional solution needs to pack fractional copies of bins with a given extended configuration, then we round up the number of such bins. The large items clearly can be packed in these bins according to the configurations of the large items. The small items are now assigned to windows (by an integral assignment), and not to specific bins. Therefore, our last steps are devoted to packing the small items.

We first place the small items which are packed in a common window type into the bins with this window as part of their extended configuration in a round-robin fashion where the small items are sorted according to their size (this ensures us that the number of items in each such bin will be approximately the same, and the total size of these items in such bins will be approximately the same). In this initial packing, some bins may be temporarily packed with a total size which exceeds 1. This excess needs to be dealt with by removing items and packing them in an alternative way. However, the excess of size of small items in a bin is relatively small (with respect to the total size of small items in this bin). In fact it is at most one excess item per bin plus a small size of additional small items (this additional small size is due to a rounding we have done when we define the set of windows). The excess items are packed in dedicated bins such that $\frac{1}{\varepsilon}$ excess items are packed in each dedicated bin. The additional items of total small size are packed again in dedicated bins such that these items from $\frac{1}{\varepsilon}$ bins are packed into one common dedicated bin. The items which are removed from a bin after the process of the round-robin allocation are the largest small items of this given excess size. The resulting scheme is an AFPTAS for GCBP, as claimed by the following theorem.

Theorem 11 *The above scheme is an AFPTAS for* GCBP.

4.3   A detailed description and analysis of the AFPTAS for GCBP

Let $0 < \varepsilon \leq \frac{1}{3}$ be such that $\frac{1}{\varepsilon}$ is an integer. Recall that $f(0) = 0$ and $f(1) = 1$.

The input for this problem includes in addition to the list of items I, also the function f. Therefore, the running time needs to be polynomial in the following three parameters: n, $\frac{1}{\varepsilon}$, and the binary representations of the numbers in the input, including the item sizes, and the values of f on the integers $1, \ldots, n$. Thus the length of the representation of f is at least $\log f(n)$.

If $n \leq \frac{1}{\varepsilon}$, we pack each item into a separate bin. In this case, the cost of the solution is at most $\frac{f(1)}{\varepsilon} = \frac{1}{\varepsilon} \leq (1 + \varepsilon)\text{OPT} + \frac{1}{\varepsilon}$. We therefore assume that $n > \frac{1}{\varepsilon}$.

Linear grouping. An item j is *large* if $s_j \geq \varepsilon$. Other items will be *small*, and would be partitioned into the *smallest* items and *medium* items. We denote by L the set of large items. We perform linear grouping of the large items. That is, if $|L| \geq \frac{1}{\varepsilon^3}$, then for $m = \frac{1}{\varepsilon^3}$ we partition L into m classes $L_1, \ldots, L_m$ such that $\lceil |L|\varepsilon^3 \rceil = |L_1| \geq |L_2| \geq \cdots \geq |L_m| = \lfloor |L|\varepsilon^3 \rfloor$, and 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 size items. For every $j = 2, 3, \ldots, m$ we round up the size of the elements of L_j to the largest size of an element of L_j. For an item i, we denote by s'_i the rounded

up size of the item. If $|L| < \frac{1}{\varepsilon^3}$, then each large item has its own set L_i, L_1 is an empty set, and for a large item j we let $s'_j = s_j$ (i.e., we do not apply rounding in this case). In both cases we have $|L_1| \leq 2\varepsilon^3 |L|$.

For items in L_1, we do not round the sizes, and we denote $s'_j = s_j$ for all $j \in L_1$. For $j \in I \setminus L$ we also let $s'_j = s_j$. We denote by $L' = L \setminus L_1$. We consider the instance I' consisting of the items in $L' \cup (I \setminus L)$ with the (rounded up) sizes s'. Then, using the standard arguments of linear grouping we conclude $\text{OPT}(I') \leq \text{OPT}(I)$. The items in L_1 are packed each in a separate bin. Let H be the set of different rounded up sizes of large items, and denote the number of items in L' with size v by $n(v)$. Note that $|H| \leq m$. We next describe the packing of the items in I'.

Dealing with the set of the small items. We define a partition of the set $I \setminus L$ of small items into two parts M and S (called medium items and smallest items, respectively), such that S is a suffix of the list of input items (i.e., a set of smallest items). Specifically, if $i \in M$ and $j \in S$, then $s'_i \geq s'_j$. Let S be a suffix $\{\eta, \ldots, n\}$ of minimum cardinality, such that $S \subseteq I \setminus L$, for which the total size exceeds $h(\varepsilon)$, where $h(\varepsilon)$ is defined as

$$h(\varepsilon) = \frac{6\ell}{\varepsilon} \cdot \left(\frac{1}{\varepsilon^3} + 1 \right)^{\frac{1}{\varepsilon}}, \tag{1}$$

and ℓ is an integer defined later, satisfying $\ell = O(\frac{1}{\varepsilon} + \log_{1+\varepsilon} f(n))$. Note that $h(\varepsilon) \geq \frac{1}{\varepsilon}$ is an integer for any valid choice of ε. If the total size of $I \setminus L$ is at most $h(\varepsilon)$ then we let $S = I \setminus L$ and $M = \emptyset$. Otherwise, the total size of the items of S is at most $h(\varepsilon) + \varepsilon$. We will pack the items from S independently from other items. That is, there are no mixed bins containing items from S and items not from S.

The first packing step of the algorithm is to pack the items of S using the following heuristic. We apply FNFI (processing the items in an order which is reverse to their order in the input). This results in $1 + h(\varepsilon)$ bins, unless $S = I \setminus L$. Afterwards, a new dedicated bin is used for every item that was split between two bins by FNFI. There are at most $h(\varepsilon)$ such items.

In order to focus on solutions that pack the items of S as we do, we next bound the cost of a solution that packs the items in S in this exact way (packed by FNFI in separate bins, where split items are moved to an additional bin). On the other hand, we relax our requirements of a solution and allow fractional packing of the items in M. The solution clearly needs to pack the items in L' as well (no fractional packing can be allowed for large items). We denote the optimal cost of such a solution by $\text{OPT}'(I')$. The motivation for allowing fractional packing of the items of M is that our goal is to bound the cost of solutions to a linear program that we introduce later, and this linear program allows fractional packing of small items that are considered by it, which are exactly the items of M (while the items of S remain packed as defined above).

Lemma 12 $\text{OPT}'(I') \leq (1 + \varepsilon)\text{OPT}(I') + (3h(\varepsilon) + 3) \cdot f(\frac{1}{\varepsilon}) \leq (1 + \varepsilon)\text{OPT}(I) + (3h(\varepsilon) + 3) \cdot f(\frac{1}{\varepsilon})$.

Proof. Consider an optimal solution $\overline{\text{OPT}}$ to the following relaxation GCBP'_ε of our packing problem. We need to pack the items of I' (with rounded up sizes) but *all* the items of $M \cup S$ can be packed fractionally. The difference with the packing $\text{OPT}'(I')$ is that items of S can be packed in an arbitrary way, and not necessarily into dedicated bins, as is described above. In particular, they can be packed fractionally. The difference with the packing $\text{OPT}(I')$ is the possibility to pack the small items fractionally. The cost of $\overline{\text{OPT}}$ is clearly at most $\text{OPT}(I') \leq \text{OPT}$.

We sort the bins of $\overline{\text{OPT}}$ in a non-increasing order, according to the number of items (i.e., the sum of fractions of items) packed in the bin (including large items). Let σ_i be the total free space in bin i

that is left after packing its large items in it. This is the space which is used by small items, together with all the free space, if exists. Let $\Sigma_i = \sum\limits_{j=1}^{i} \sigma_i$. Let $p = \min\{i | \Sigma_i \geq \sum\limits_{j \in S} s'_j\}$. The integer p must exist since all items of S must be packed.

We show that without loss of generality, we can assume that all items of S are packed in bins $1, 2, \ldots, p$ in $\overline{\text{OPT}}$. To show this, consider an optimal solution to GCBP'_ε that minimizes the following function (among all optimal solutions): the number of existing quadruples (a_1, i_1, a_2, i_2), where $a_1 \leq p < a_2$, $i_1 \in M$, $i_2 \in S$, and there is a non-zero fraction of item i_j packed in bin a_j, for $j = 1, 2$. Assume by contradiction that such a quadruple (a_1, i_1, a_2, i_2) exists. Let γ be the fraction of i_1 in bin a_1 and δ the fraction of i_2 in bin a_2.

Let $\mu = \min\{\gamma \cdot s_{i_1}, \delta \cdot s_{i_2}\}$. Denote the fractions of i_1 and i_2 of size μ by $\gamma' = \frac{\mu}{s_{i_1}}$ and $\delta' = \frac{\mu}{s_{i_2}}$. We swap a part of size μ of item i_2 in bin a_2 with a part of size μ of item i_1 in bin a_1. Since $s_{i_1} \geq s_{i_2}$ (recall that S contains the smallest items), we get that the fractions satisfy $\gamma' \leq \delta'$. The number of items in bin a_1 was increased by $\delta' - \gamma'$, and in bin a_2 it was decreased by $\delta' - \gamma'$. The sorted order of bins may have changed as a result, but bin a_1 can be moved to an earlier spot while a_2 may be moved to a later spot, so the set of the first p bins does not change. Moreover, we destroyed at least one quadruple, and did not create new ones, since no parts of items of M were moved to bins $1, \ldots, p$ and no items of S were moved to bins $p+1, p+2, \ldots$. Let n_1 and n_2 be the numbers of items in bins a_1 and a_2 before the change. The change in the cost function is $f(n_1 + \delta' - \gamma') + f(n_2 - (\delta' - \gamma')) - f(n_1) - f(n_2) \leq 0$, since $n_1 \geq n_2$, $\delta' - \gamma' \geq 0$, and by concavity. Therefore, the resulting solution has a cost of at most $\overline{\text{OPT}}$, and the minimality is contradicted.

If no such quadruple exists then there are two cases. If all bins $p+1, p+2, \ldots$ contain only fractions of items of M (possibly in addition to large items), then all items of S are in bins $1, \ldots, p$ and our assumption holds. Otherwise, we have that all bins $1, \ldots, p$ contain no fractions of items in M. In this case, if there are items of S in any of the bins $p+1, p+2, \ldots$, then there must be empty space in bins $1, \ldots, p$. Parts of items of S can be repeatedly moved to these bins, until no parts of items of S exist in bins $p+1, p+2, \ldots$. In each such step, the number of items in some bin in $1, \ldots, p$ increases, and the number of items in some bin in $p+1, p+2, \ldots$ decreases. Sorting the bins again after every such step (according to a non-increasing numbers of items) will contain the same set of bins in the prefix of p bins, and our assumption holds as well. Due to concavity, and since the target bin cannot contain less items than the source bin, every such step cannot increase the cost.

We next adapt $\overline{\text{OPT}}$ by creating at most $h(\varepsilon) + 2$ additional bins, and pack the small items of the first p bins into these bins using FNFI. By the definition of p, the total size of items of S packed into the first p bins is at most $h(\varepsilon) + \varepsilon$. The p-th bin may contain items of M with a total size of less than $\sigma_p \leq 1$, where the last item of M packed into the p-th bin may be packed fractionally, with its second part packed into the $(p+1)$-th bin. Thus, this set of small items may contain items of M of total size at most $1 + \varepsilon$, and the total size of small items which are packed (possibly partially) into the first p bins is at most $h(\varepsilon) + 1 + 2\varepsilon < h(\varepsilon) + 2$. We denote this set of items that is moved by $\hat{S}$. We compute the change in the cost and afterwards adapt the solution further so that it complies with the requirement that the items of S are packed integrally in separate bins, as is done in OPT'.

We define an auxiliary monotonically non-decreasing concave function $\tilde{f}$ as follows. $\tilde{f}(x) = f(x + \frac{1}{\varepsilon}) - f(\frac{1}{\varepsilon})$. Note that $\tilde{f}(0) = 0$. Consider the p bins of $\overline{\text{OPT}}$ from which the small items are removed. Let r_i and a_i denote the numbers of large and small items in these original bins. Clearly, $r_i \leq \frac{1}{\varepsilon}$. By removing the small items, the cost of such a bin decreases by $f(a_i + r_i) - f(r_i) \geq f(a_i + \frac{1}{\varepsilon}) - f(\frac{1}{\varepsilon}) = \tilde{f}(a_i)$, where the inequality is due to concavity of f. For every bin which is

created for small items, if it contains b_i small items, its cost is $f(b_i) \leq f(b_i + \frac{1}{\varepsilon}) = \tilde{f}(b_i) + f(\frac{1}{\varepsilon})$, where the inequality is due to monotonicity.

Consider now the packing of the items $\hat{S}$ that is implied by the solution $\overline{\text{OPT}}$, with respect to the function $\tilde{f}$, and neglecting the large items. The cost of this packing for bin i is $\tilde{f}(a_i)$. Let $\tilde{A}$ denote the total cost of all the bins that contain items of $\hat{S}$, that is, of the first p bins. Let $\tilde{B}$ denote the total cost with respect to $\tilde{f}$ of all the bins that are created by FNFI for $\hat{S}$. In this case the cost of a bin i is $\tilde{f}(b_i)$. That is, $\tilde{A} = \sum\limits_{i=1}^{p} \tilde{f}(a_i)$ and $\tilde{B} = \sum\limits_{i=1}^{h(\varepsilon)+2} \tilde{f}(b_i)$. By Lemma 10 (that holds even though the value $\tilde{f}(1)$ can be arbitrary), we have $\tilde{A} \geq \tilde{B}$.

Let Δ denote the difference in the cost for the items of $\hat{S}$. We have $\Delta = \sum\limits_{i=1}^{h(\varepsilon)+2} f(b_i) - \sum\limits_{i=1}^{p}(f(r_i + a_i) - f(r_i)) \leq \sum\limits_{i=1}^{h(\varepsilon)+2}(\tilde{f}(b_i) + f(\frac{1}{\varepsilon})) - \sum\limits_{i=1}^{p} \tilde{f}(a_i) \leq (h(\varepsilon) + 2)f(\frac{1}{\varepsilon})$ (by the previous claims and $\tilde{A} \geq \tilde{B}$).

We next convert the packing of $\hat{S}$ as follows. If there exists a mixed bin, that is, a bin containing items from both S and M, we split it into two bins, so that the two subsets of M and of S are separated. If a mixed bin indeed exists, $M \neq \emptyset$, and the total size of the S items is more than $h(\varepsilon)$, but not more than $h(\varepsilon) + 1$. Therefore, the split bin appears as the $h(\varepsilon) + 1$-th bin created by FNFI. Moreover, the number of items in the $h(\varepsilon) + 1$-th bin is no larger than the number of items in every earlier bin. Therefore, if the number of items in the $h(\varepsilon) + 1$-th bin is N, then the current cost is at least $f(N)(h(\varepsilon) + 1)$ and as a result of the split, the cost increases by an additive factor of at most $f(N)$. So the multiplicative factor of the increase in the cost is at most $1 + \frac{1}{h(\varepsilon)+1} \leq 1 + \varepsilon$ where the inequality holds by $h(\varepsilon) \geq \frac{1}{\varepsilon}$.

For a pair of consecutive bins created by FNFI (excluding at most two bins with items of $\hat{S} \cap M$), if an item was split between the two bins, it is removed from these bins and packed completely in a new bin dedicated to it. In addition, there may be at most one item of M which was split between bins and if it exists, it is moved into a dedicated bin as well. There are at most $h(\varepsilon) + 1$ items that may have been moved into dedicated bins, so this increases the cost by at most $(h(\varepsilon) + 1) \cdot f(1)$. At this time, the items of S are packed exactly as in $\text{OPT}'(I')$.

The total cost is at most

$$
\begin{aligned}
(1 + \varepsilon)(\overline{\text{OPT}} + (h(\varepsilon) + 2)f(\frac{1}{\varepsilon})) + h(\varepsilon) + 1 &\leq (1 + \varepsilon) \cdot \overline{\text{OPT}} + ((2 + \varepsilon)h(\varepsilon) + 3 + 2\varepsilon)f(\frac{1}{\varepsilon}) \\
&\leq (1 + \varepsilon) \cdot \overline{\text{OPT}} + (3h(\varepsilon) + 3)f(\frac{1}{\varepsilon})
\end{aligned}
$$

(using $\varepsilon \leq \frac{1}{3}$ and $h(\varepsilon) \geq 1$). $\blacksquare$

We next need to pack the items in $I'' = I' \setminus S = M \cup L'$. In the case where I'' is empty, the proof of the next corollary follows from Lemma 12. We will show at the end of this section that this results in an AFPTAS for the case $I'' = \emptyset$.

Corollary 13 *If I'' is empty, then the last lemma shows that the resulting solution costs at most $(3h(\varepsilon) + 3)f(\frac{1}{\varepsilon})$.*

We next consider the case $I'' \neq \emptyset$.

Definition 3 *Let $s_{min} = \min\limits_{i \in I''} s_i'$. Let $\boldsymbol{\Delta} = \frac{1}{s_{min}}$ if $M \neq \emptyset$ and otherwise $\boldsymbol{\Delta} = \frac{1}{\varepsilon}$, that is, $\boldsymbol{\Delta} = \frac{1}{\min\{s_{min}, \varepsilon\}}$. Clearly, for any $i \in S$ we have $s_i' \leq s_{min}$ and $s_i' \leq \frac{1}{\boldsymbol{\Delta}}$.*

Consider the instance I''. In the temporary solutions, we allow fractional packing of the items of M and we use $\text{OPT}'(I'')$ to denote an optimal packing of I'' where small items may be packed fractionally. This does not change the fact that any bin, packed with items of a total size of at most 1, can contain a total number of items of at most $\mathbf{\Delta}$ even if it contains fractions of items.

Definition 4 *We denote the cost of the bins packed with the items of S by $F(S)$, that is, $\text{OPT}'(I') = \text{OPT}'(I'') + F(S)$.*

The items of S, if packed by FNFI (which by Lemma 10 is a minimum cost packing for them) require at least $h(\varepsilon)$ full bins, with at least $\mathbf{\Delta}$ items in each. Therefore, we have

$$F(S) \geq h(\varepsilon) \cdot f(\mathbf{\Delta}). \tag{2}$$

On the other hand, at this time, any other valid bin can contain a total number of items of at most $\mathbf{\Delta}$.

These properties are true unless $M = \emptyset$. In that case, only large items remain to be packed, so the number of items in any additional bin is at most $\frac{1}{\varepsilon}$.

Approximating the cost function f. Given the function f we compute a staircase function, which is an $(1+\varepsilon)$-approximation of f from below, with $O(\frac{1}{\varepsilon} + \log_{1+\varepsilon} f(n))$ breakpoints. That is, we find a sequence of integers $0 = k_0 < k_1 = 1 < \cdots < k_{\frac{1}{\varepsilon}} = \frac{1}{\varepsilon} < k_{\frac{1}{\varepsilon}+1} < \cdots < k_\ell = n$ such that for all $i = \frac{1}{\varepsilon}, \frac{1}{\varepsilon}+1, \ldots, \ell-1$, we have $f(k_{i+1}) \leq (1+\varepsilon)f(k_i)$. The sequence is constructed as follows. We define $k_j = j$ for $j = 0, 1, \ldots, \frac{1}{\varepsilon}$. Every subsequent value k_{j+1} for $j \geq \frac{1}{\varepsilon}$ is defined as the maximum integer $t > k_j$ such that $f(t) \leq (1+\varepsilon)f(k_j)$. Note that this definition is valid since for $\zeta \geq \frac{1}{\varepsilon}$ we have $f(\zeta+1) \leq f((1+\varepsilon)\zeta) \leq (1+\varepsilon)f(\zeta)$, where the first inequality holds by the monotonicity of f, and the second inequality holds by the concavity of f. Then, by the definition of the sequence, for every $i = \frac{1}{\varepsilon}, \frac{1}{\varepsilon}+1, \ldots, \ell-2$, we have $f(k_{i+2}) > (1+\varepsilon)f(k_i)$. This implies

$$\ell \leq \frac{1}{\varepsilon} + 1 + 2\log_{1+\varepsilon} f(n) \leq \frac{1}{\varepsilon} + 1 + 2\log_{1+\varepsilon} \text{OPT}(I) \tag{3}$$

and $\ell \leq n$. Let Δ be such that $k_\Delta \geq \mathbf{\Delta}$ and $k_{\Delta-1} \leq \mathbf{\Delta}$. If $M = \emptyset$, we have $\mathbf{\Delta} = \frac{1}{\varepsilon}$, so $k_\Delta = \frac{1}{\varepsilon}$. The staircase function, which is an $(1+\varepsilon)$-approximation of f from below, is defined as the value of f for values k_i, and it remains constant between these points. Using these definitions, and (2) we get

$$F(S) \geq h(\varepsilon)f(k_{\Delta-1}) \geq \frac{h(\varepsilon)}{1+\varepsilon}f(k_\Delta) \, . \tag{4}$$

Configurations and windows. Given the instance I'', we define configurations.

Definition 5 *A configuration of a bin C is a (possibly empty) set of items of L' whose total (rounded up) size is at most 1. The set of all configurations is denoted by $\mathring{\mathcal{C}}$.*

Since L' contains only $|H|$ types of items, a configuration can be viewed as a multiset of items of H. We next define n_C for a configuration C, and $NUMBER(v, C)$ for a pair of an item type in H and a configuration C.

Definition 6 *For each $v \in H$, and $C \in \mathring{\mathcal{C}}$, let $NUMBER(v, C)$ denote the number of items of size v in configuration C. We use n_C to denote the number of large items in C, that is,*

$$n_C = \sum_{v \in H} NUMBER(v, C) \, .$$

18

For a configuration C we define an additional value, $\tau(C)$, which is an approximation of the available size for small items in a bin with configuration C.

Definition 7 *Let the total (rounded up) size of the items in C be denoted by $s'(C)$. We let $\tau(C) = \frac{1}{(1+\varepsilon)^t}$ where t is the maximum integer such that $0 \le t \le \log_{1+\varepsilon} \frac{1}{s'_{min}} + 1$ and $s'(C) + \frac{1}{(1+\varepsilon)^t} \ge 1$.*

For each configuration C we define $\Delta + 1 \le \ell + 1$ *extended configurations* (C, i), where $0 \le i \le \Delta$. A bin packed according to an extended configuration (C, i) has large items according to configuration C, and at most k_i items in total (that are either large or small items, i.e., including the large items of this configuration). We later slightly relax this condition and allow to increase the number of items in a bin (in favor of possibly packing a slightly larger number of small items) in a way that the cost of this bin only increases by a factor of $1 + \varepsilon$.

Definition 8 *An extended configuration (C, p), where $C \in \mathring{\mathcal{C}}$ and $1 \le p \le \Delta$, is called valid or feasible if $n_C \le k_p$. Denote the set of all extended feasible configurations by $\mathcal{C}$.*

We use the notation $n(C, p)$ as an upper bound on the total number of small items that can fit into a bin packed with the extended configuration (C, p).

Definition 9 *The value $n(C, p)$ is defined as follows. Let t be the smallest integer such that $k_p - n_C \le k_t$. Then we define $n(C, p) = t$.*

Recall that the minimum size of an item is denoted by $s_{min} = \min_{i \in M \cup L'} s'_i$ (note that $s_{min} \neq 0$), and we let $s'_{min} = \max\{\frac{1}{(1+\varepsilon)^t} | t \in \mathbb{Z}, \frac{1}{(1+\varepsilon)^t} \le s_{min}\}$ be an approximated value of s_{min} which is an integer power of $1 + \varepsilon$. The value $\log_{1+\varepsilon} \frac{1}{s'_{min}}$ is polynomial in the size of the input and in $\frac{1}{\varepsilon}$.

Definition 10 *Let the set*

$$\mathcal{W} = \{(\frac{1}{(1 + \varepsilon)^t}, a) | 0 \le t \le \log_{1+\varepsilon} \frac{1}{s'_{min}} + 1, 0 \le a \le \ell\}$$

be called the set of all possible windows. A window is defined as a member of $\mathcal{W}$. For two windows, W^1 and W^2 where $W^i = (\omega_i, a_i)$ for $i = 1, 2$, we say that $W^1 \le W^2$ if $\omega_1 \le \omega_2$ and $a_1 \le a_2$.

The intuitive meaning of a window here is a pair consisting of a bound on the remaining capacity for small items in a bin (this bound is rounded to an integer power of $1 + \varepsilon$), and an upper bound k_a on the number of small items packed into a bin. Then, $|\mathcal{W}| \le (\ell + 1) \cdot (\log_{1+\varepsilon} \frac{1}{s'_{min}} + 2)$.

Note that each bin that contains large items, packed according to an extended configuration (C, p), may leave space for small items.

Definition 11 *For an extended configuration (C, p) we define the main window of (C, p) to be*

$$MAIN(C, p) = (\tau(C), n(C, p)),$$

where $\tau(C)$ is defined in Definition 7 and $n(C, p)$ is defined in Definition 9.

Corollary 14 *Given an extended configuration (C, p), the real cost (after adding small items such that their number is not larger than the number in the main window of (C, p), i.e., $n(C, p)$) of a bin that is packed according to this extended configuration, is at most $(1 + \varepsilon)f(k_p)$.*

19

Proof. Recall that $n(C, p) = k_t$ if t be the smallest integer such that $k_p - n_C \leq k_t$. It can be seen that $t \leq p$ always holds.

If $k_p - n_C = k_t$, then $f(n(C, p) + n_C) \leq (1 + \varepsilon)f(k_p)$ clearly holds. Otherwise, $k_p - n_C \neq k_t$ holds and then $k_t > \frac{1}{\varepsilon}$, and $t > \frac{1}{\varepsilon}$, so we have $k_p - n_C > k_{t-1}$. Hence in this case we conclude that $f(n(C, p) + n_C) = f(k_t + n_C) \leq f(k_t + k_p - k_{t-1}) \leq f(k_p) + f(k_t) - f(k_{t-1}) \leq f(k_p) + \varepsilon f(k_{t-1}) \leq (1 + \varepsilon)f(k_p)$, where the first inequality holds by the definition of t and the monotonicity of f, the second inequality holds by the concavity of f (since $k_p \geq k_t > k_{t-1}$ and $k_p < k_p + k_t - k_{t-1}$ hold, $f(k_t) + f(k_p) \geq f(k_t + k_p - k_{t-1}) + f(k_{t-1})$), the third inequality holds because $f(k_t) \leq (1 + \varepsilon)f(k_{t-1})$ and the last inequality holds by the monotonicity of f and since $t - 1 < p$ which implies $k_{t-1} < k_p$. $\blacksquare$

The main window of an extended configuration is a window (i.e., it belongs to $\mathcal{W}$), but $\mathcal{W}$ may include windows that are not the main window of any extended configuration. We note that $|\mathcal{W}|$ is polynomial in the input size and in $\frac{1}{\varepsilon}$, whereas $|\mathcal{C}|$ may be exponential in $\frac{1}{\varepsilon}$ (specifically, $|\mathcal{C}| \leq \ell \cdot (\frac{1}{\varepsilon^3} + 1)^{1/\varepsilon}$).

Definition 12 *Denote the set of windows that are main windows of at least one extended configuration by $\mathcal{W}'$.*

We first define a linear program that allows the usage of any window in $\mathcal{W}$. After we obtain a solution to this linear program, we modify it so that it only uses windows of $\mathcal{W}'$.

Definition 13 *A generalized configuration G is a pair of pairs $G = ((C, p), W = (\omega, j))$, for some feasible extended configuration (C, p) and some $W \in \mathcal{W}$. The generalized configuration G is valid if $W \leq MAIN(C, p)$. The set of all valid generalized configurations is denoted by $\mathcal{G}$. For $W \in \mathcal{W}$ denote by $G(W)$ the set of valid generalized configurations $G = ((C, p), W')$ such that W is their window, i.e., $G(W) = \{((C, p), W') \in \mathcal{G} : W' = W\}$.*

The linear program. We next consider the following linear program. The program is examined but it is not constructed explicitly by the algorithm. In this linear program we have a variable x_G denoting the number of bins with generalized configuration G, and variables $y_{i,W}$ indicating if the small item i is packed in a window of type W (the exact instance of this window is not specified in a solution of the linear program).

$$\min \quad \sum_{G=((C,p),W)\in\mathcal{G}} f(k_p)x_G$$

$$s.t. \quad \sum_{G=((C,t),W)\in\mathcal{G}} NUMBER(v, C)x_G \geq n(v) \quad \forall v \in H \tag{5}$$

$$\sum_{W\in\mathcal{W}} y_{i,W} \geq 1 \quad \forall i \in M \tag{6}$$

$$\omega \cdot \sum_{G\in G(W)} x_G \geq \sum_{i\in M} s'_i \cdot y_{i,W} \quad \forall W = (\omega, p) \in \mathcal{W} \tag{7}$$

$$k_p \cdot \sum_{G\in G(W)} x_G \geq \sum_{i\in M} y_{i,W} \quad \forall W = (\omega, p) \in \mathcal{W} \tag{8}$$

$$x_G \geq 0 \quad \forall G \in \mathcal{G}$$

$$y_{i,W} \geq 0 \quad \forall W \in \mathcal{W}, \forall i \in M.$$

Constraints (5) and (6) ensure that each item (large or small) of I'' will be considered. The large items will be packed by the solution, and the small items would be assigned to some type of window. Constraints (7) ensure that the total size of the small items that we decide to pack in window of type W is not larger than the total available size in all the bins that are packed according to a generalized configuration, whose window is of type W (according to the window size). Similarly, the family of constraints (8) ensures that the total number of the small items that we decide to pack in a window of type W is not larger than the total number of small items that can be packed (in accord with the second component of W) in all the bins whose generalized configuration of large items induces a window of type W. In what follows we show how to deal with small items and specifically, how to pack most of them into the windows allocated for them, and how to further deal with some unpacked small items.

Lemma 15 *There is a feasible solution to the above linear program that has a cost of at most $(1 + \varepsilon)\mathrm{OPT}'(I'')$.*

Proof. The $(1 + \varepsilon)$ factor results from the fact that we define extended configurations, where the number of items per bin is k_p (for some value of p). The fact that we use a window (ω, t), where k_t belongs to the same sequence of values, k_i, will result in an additional factor of $1 + \varepsilon$ on the cost of the linear program.

We now convert the solution given by $\mathrm{OPT}'(I'')$. To convert the solution, we do not need to modify any packing of items, but we change the cost calculation of each bin to comply with costs of generalized configurations. For this, the number of items in every bin must be converted (in favor of cost calculations) as follows.

Given a bin with $n_1 > 0$ items, we define p to be minimal value such that $k_p \geq n_1$. An increase in the cost can occur if $k_p > n_1$. In this case, $p > 0$ and we have $k_{p-1} < n_1 < k_p$ and thus using monotonicity of f and the properties of the sequence k_i we have $f(k_p) \leq (1 + \varepsilon)f(k_{p-1}) \leq (1+\varepsilon)f(n_1)$. Since windows are never smaller than the real space in bins, both with respect to size and with respect to the difference between the number of large items and the value k_p of the configuration, the solution clearly satisfies the constraints (7) and (8) on the packing of small items, and the packing of large items satisfies the constraints (5). Therefore the adapted solution is a feasible solution of the linear program. The linear program calculates the cost of a packing using the values k_p of the extended configurations, and as shown above, this increases the cost of $\mathrm{OPT}'(I'')$ by a multiplicative factor of at most $1 + \varepsilon$. ∎

Note that in the proof of Lemma 15, the presented solution to the linear program is such that any variable x_G, that corresponds to generalized configurations $G = ((C, p), W)$ for which $W \neq MAIN(C, p)$, is equal to zero, and any variable $y_{i,W}$ where $W \notin \mathcal{W}'$ is equal to zero as well.

The column generation technique. We invoke the column generation technique of Karmarkar and Karp [22] as follows. The above linear program may have an exponential number of variables and 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) that we describe next.

The dual variables α_v correspond to the item sizes in H, and the dual variables β_i correspond to the small items of M. The intuitive meaning of these two types of variables can be seen as weights of these items. For each $W \in \mathcal{W}$ we have a pair of dual variables γ_W, δ_W. Using these dual variables, the dual linear program is as follows. Once again, the program is examined but not constructed explicitly by the algorithm.

$$\max \qquad \sum_{v \in H} n(v)\alpha_v + \sum_{i \in M} \beta_i$$

$$s.t. \quad \sum_{v \in H} NUMBER(v,C)\alpha_v + \omega\gamma_W + k_t\delta_W \leq f(k_p) \quad \forall G = ((C,p), W = (\omega,t)) \in \mathcal{G} \ (9)$$

$$\beta_i - s_i'\gamma_W - \delta_W \leq 0 \qquad\qquad \forall i \in M, \forall W \in \mathcal{W} \qquad\qquad (10)$$

$$\alpha_v \geq 0 \qquad\qquad\qquad \forall v \in H$$

$$\beta_i \geq 0 \qquad\qquad\qquad \forall i \in M$$

$$\gamma_W, \delta_W \geq 0 \qquad\qquad\qquad \forall W \in \mathcal{W}.$$

First note that there is a polynomial number of constraints of type (10), and therefore we clearly have a polynomial time separation oracle for these constraints. If we would like to solve the above dual linear program (exactly) then using the ellipsoid method we need to establish the existence of a polynomial time separation oracle for the constraints (9). However, we are willing to settle on an approximated solution to this dual program. 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, $\frac{1}{\varepsilon}$, $\log \frac{1}{s'_{min}}$ and $\log f(n)$) such that for a given solution $a^* = (\alpha^*, \beta^*, \gamma^*, \delta^*)$ decides whether a^* is a feasible dual solution (approximately). That is, it either provides a generalized configuration $G = ((C,p), W = (\omega,t)) \in \mathcal{G}$ for which $\sum_{v \in H} NUMBER(v,C)\alpha_v^* + \omega\gamma_W^* + k_t\delta_W^* > f(k_p)$, or outputs that an approximate infeasibility evidence does not exist, that is, for all generalized configurations $G = ((C,p), W = (\omega,t)) \in \mathcal{G}$, $\sum_{v \in H} NUMBER(v,C)\alpha_v^* + \omega\gamma_W^* + k_t\delta_W^* \leq (1 + \varepsilon)f(k_p)$ holds. In such a case, we will show that $\frac{a^*}{1+\varepsilon}$ is a feasible dual solution which also satisfies constraints (10), that can be used.

An algorithm for finding approximate infeasibility evidence, using a reduction to a knapsack problem. Our algorithm for finding an approximate infeasibility evidence uses the following problem as an auxiliary problem. The KNAPSACK PROBLEM WITH A MAXIMUM CARDINALITY CONSTRAINT (KCC) problem is defined as follows. Given a set of item types H and an integer value k, where each item type $v \in H$ has a given multiplicity $n(v)$, a value z_v^* and a size v, the goal is to pack a multiset of at most k items (taking the multiplicity, in which items are taken, into account, and letting the solution contain at most $n(v)$ items of type v) and a total size of at most 1, so that the total value is maximized. To provide an FPTAS for KCC, note that one can replace an item with size v by $n(v)$ copies of this item and then one can apply the FPTAS of Caprara et al. [8] for the knapsack problem with cardinality constraints. The FPTAS of [8] clearly has polynomial time in the size of its input, and $\frac{1}{\varepsilon}$. Since the number of items that we give to this algorithm as input is at most n, we can use this FPTAS and still let our scheme have polynomial running time.

A configuration G, that is an approximate infeasibility evidence, can be found by the following procedure: For each $W = (\omega,t) \in \mathcal{W}$, and for every $0 \leq p \leq \ell$, we look for an extended configuration $(C,p) \in \mathcal{C}$ such that $((C,p), W)$ is a valid generalized configuration, and such that $\sum_{v \in H} NUMBER(v,C)\alpha_v^*$ is maximized. To find C, we invoke the FPTAS for the KCC problem with the following input: The set of items is H where for each $v \in H$ there is a value α_v^* and a size v, the goal is to pack a multiset of the items, so that the total value is maximized, under the following conditions. The multiset should consist of at most $k_p - k_{t-1} - 1$ large items, (taking the multiplicity into account, but an item can appear at most a given number of times). The rounded up number of

small items which are packed in (C, p) is k_t, and therefore their number is in the interval $[k_{t-1}+1, k_t]$. Therefore, the number of large item in such an extended configuration is always at most $k_p - k_{t-1} - 1$. By allowing this larger number of large items we only relax the constraints, and hence will get super optimal solutions which will be sufficient.

If $t = 0$, we instead search for a multiset with at most k_p large items. The total (rounded up) size of the multiset should be smaller than $1 - \frac{\omega}{1+\varepsilon}$, unless $\omega < s'_{min}$, where the total size should be at most 1 (in this case, the window does not leave space for small items). Since the number of applications of the FPTAS for the KCC problem is polynomial (i.e., $(\ell + 1)|\mathcal{W}|$), this algorithm runs in polynomial time.

The case where a solution with large value is found. In the case that a solution is found, that is, a configuration C, with at most $k_p - k_{t-1} - 1$ large items (or k_p, if $t = 0$), and a total value greater than $f(k_p) - \omega \gamma_W^* - k_t \delta_W^*$, we argue that $((C, p), (\omega, t))$ is indeed a valid generalized configuration, and this implies that there exists a generalized configuration, whose dual constraint (9) is violated. First, we need to show that (C, p) is a valid extended configuration. This holds since C has at most $k_p - k_{t-1} - 1 \leq k_p$ large items (if $t = 0$ the bound on the number of items holds immediately).

By the definition of windows, the property $\omega < s'_{min}$ is equivalent to $\omega = \frac{s'_{min}}{1+\varepsilon}$, which is the smallest size of window (and the smallest sized window forms a valid generalized configuration with any configuration, provided that the value of k_t is small enough). If $t > 0$, since C has at most $k_p - k_{t-1} - 1$ items, the second component of the main window of C in this case is larger than $t - 1$ and thus no smaller than t, and the window is no smaller than (ω, t). Therefore, the generalized configuration $((C, p), (\omega, t))$ is valid. If $t = 0$ then the window $(\omega, 0)$ is clearly valid with any extended configuration (for the current value of ω).

If $\omega \geq s'_{min}$, recall that the main window of (C, p), $MAIN(C, p) = (\tau(C), n(C, p))$ is chosen so that $s'(C) + \tau(C) \geq 1$, and that C is chosen by the algorithm for KCC so that $s'(C) < 1 - \frac{\omega}{1+\varepsilon}$. We get $1 - \tau(C) \leq s'(C) < 1 - \frac{\omega}{1+\varepsilon}$ and therefore $\omega < (1 + \varepsilon)\tau(C)$, i.e., $\omega \leq \tau(C)$ (since the sizes of windows are integer powers of $1 + \varepsilon$). Since C contains at most $k_p - k_{t-1} - 1$ items, we have $n(C, p) \geq k_t$ and so we conclude that $W \leq MAIN(C, p)$, and $((C, p), W)$ is a valid generalized configuration (the same property holds for $t = 0$). Thus in this case we found that this solution is a configuration whose constraint in the dual linear program is not satisfied, and we can continue with the application of the ellipsoid algorithm.

The case where a solution with large value is not found: Constructing a feasible dual solution. In the case that a solution is not found, it holds for any pair of a window $W = (\omega, t)$, and a value $0 \leq p \leq \ell$, that any configuration C of total rounded up size less than $1 - \frac{\omega}{1+\varepsilon}$ (or at most 1, if $\omega < s'_{min}$), with at most $k_p - k_{t-1} - 1$ items, has a value of at most $(1 + \varepsilon)(f(k_p) - \omega \gamma_W^* - k_t \delta_W^*) \leq (1 + \varepsilon)f(k_p) - \omega \gamma_W^* - k_t \delta_W^*$. We prove that in this case, all the constraints of the dual linear program are satisfied by the solution $\frac{a^*}{1+\varepsilon}$. Consider an arbitrary valid generalized configuration $G = ((C, p), (\tilde{\omega}, j))$. We have $(\tilde{\omega}, j) \leq MAIN(C, p)$. If $\tau(C) < s'_{min}$, then $\tilde{\omega} = \tau(C)$. Since $s'(C) \leq 1$ for any configuration, and $k_j \leq n(C, p)$, we prove that the number of items in C is at most $k_p - k_{j-1} - 1$ (if $j = 0$ then the number of items in C is immediately at most k_p and there is nothing to prove). Assume by contradiction that the number of items in C is at least $k_p - k_{j-1}$. Then by definition, we have $n(C, p) \leq k_{j-1}$, which is impossible. Thus, (C, p) is a possible extended configuration to be used with the window $(\tilde{\omega}, j)$ in the application of the FPTAS for KCC, or equivalently C is a possible configuration to be used with the parameter p and the window $(\tilde{\omega}, j)$ in the application of the FPTAS for KCC. Assume next that $\tilde{\omega} < 1$, then when the FPTAS for KCC is applied on $W = (\tilde{\omega}, j)$, C

23

is a configuration that is taken into account for W since $s'(C) < 1 - \frac{\tau(C)}{1+\varepsilon} \leq 1 - \frac{\tilde{\omega}}{1+\varepsilon}$, where the first inequality holds by definition of $\tau(C)$, and C has at most $k_p - k_{j-1} - 1$ items. If $\tilde{\omega} = 1$ then $1 \geq \tau(C) \geq \tilde{\omega} = 1$, so $\tau(C) = 1$. A configuration C_1 that contains at least one large item satisfies $s'(C_1) \geq \varepsilon$, so $s'(C_1) + \frac{1}{1+\varepsilon} \geq \frac{1+\varepsilon+\varepsilon^2}{1+\varepsilon} > 1$. Therefore if the main window of a configuration has size 1 (its first component), this configuration is empty, and $s'(C) = 0$. In this case, the extended configuration (C,p) is valid for any $0 \leq p \leq \ell$. We have $n(C,p) = k_p$ for the empty configuration, and for any $1 \leq j \leq p$, $k_p - k_{j-1} - 1 \geq 0$, and for $j = 0$, $k_p \geq 0$. This empty configuration is considered with any window $W = (\omega, j) \in \mathcal{W}$ where $j > 0$ in the application of KCC. Note that if $j = 0$, the configuration has no items at all (large or small).

The cost of the approximate primal solution. We denote by (X, Y) the solution to the primal linear program that we obtained. Since the number of variables is exponential, we use a short representation of this sparse vector, where we list pairs of non-zero components and their values.

Lemma 16 *The cost of (X, Y) is at most $(1 + \varepsilon)^2 \text{OPT}'(I'')$.*

Proof. The solution (X, Y) is a $(1 + \varepsilon)$ approximation for the optimal solution to the linear program. By Lemma 15, there exists a feasible solution to the primal linear program with a cost of at most $(1 + \varepsilon)\text{OPT}'(I'')$. We conclude that $\sum\limits_{G=((C,p),(\omega,t))\in\mathcal{G}} f(k_p)X_G \leq (1 + \varepsilon)^2 \text{OPT}'(I'')$. ∎

Modifying the solution to the linear program so that all windows in $\mathcal{W} \setminus \mathcal{W}'$ can be neglected. We modify the solution (X, Y) to the primal linear program into a different feasible solution of the linear program, without increasing the goal function. Using the short representation of X of the non-zero components of this vector, we create a list of generalized configurations whose X component is positive. From this list of generalized configurations, we find a list of windows that are the main window of at least one extended configuration induced by a generalized configuration in the list. This list of windows is a subset of $\mathcal{W}'$ defined in Definition 12. We would like the solution to use only windows from $\mathcal{W}'$.

The new solution will have the property that any non-zero component of X, X_G, corresponds to a generalized configuration $G = ((C,p), W)$, such that $W \in \mathcal{W}'$. We still allow generalized configurations $G = ((C,p), W)$ where $W \neq MAIN(C,p)$, as long as $W \in \mathcal{W}'$. This is done in the following way. We first let $(X', Y') = (X, Y)$. Since the vector (X, Y) will be modified, we will use the copy (X', Y') to recall the original values of the solution to the linear program.

Given a window $V \notin \mathcal{W}'$, we define $BINS(V) = \sum\limits_{\hat{G}\in G(V)} X'_{\hat{G}}$. The following is done for every generalized configuration $G' = ((C,p), V)$, where $V \notin \mathcal{W}'$ and such that $X'_{G'} > 0$, where $W = MAIN(C,p)$ and $W \geq V$ (but $V \neq W$). We let $G = ((C,p), W)$. For every $i \in M$, an amount of $\frac{X'_{G'}}{BINS(V)}Y'_{i,V}$ is transferred from $Y_{i,V}$ to $Y_{i,W}$. We modify the values $X_{G'}$ and X_G as follows. We increase the value of X_G by $X'_{G'}$ and let $X_{G'} = 0$.

We claim that at the end of the modification process, it holds that $Y_{i,V} = 0$, for any $i \in M$ and $V \notin \mathcal{W}'$. To see this, we note that the total reduction in $Y_{i,V}$ is $\sum\limits_{G'\in G(V)} \frac{X'_{G'}}{BINS(V)}Y'_{i,V} = Y'_{i,V}$, where the last equality holds by the definition of $BINS(V)$.

Lemma 17 *The new vector (X, Y) is a feasible solution to the linear program with the same value of the objective function.*

Proof. We consider the modifications. For every extended configuration (C,p), the sum of components of X, that correspond to generalized configurations whose extended configuration of large items is (C,p), does not change. Therefore, the value of the objective function is the same, and the constraints (5) still hold. We next consider the constraint (6) for i, for a given small item $i \in M$. Since the sum of variables $Y_{i,W}$ does not change, this constraint still holds.

As for constraints (7) and (8), for a window $V \notin \mathcal{W}'$, the right hand side of each such constraint became zero. On the other hand, for windows in $\mathcal{W}'$, every increase in some variable X_G for $G = ((C,p), W = (\omega, t))$, that is originated in a decrease of $X_{G'}$ for $G = ((C,p), V = (\omega', t'))$ is accompanied with an increase of $\dfrac{X'_{G'}}{\sum_{\hat{G} \in G(V)} X'_{\hat{G}}} Y_{i,V} = \dfrac{X'_{G'}}{BINS(V)} Y_{i,V}$ in $Y_{i,W}$, for every $i \in M$. This results in an increase of $\sum_{i \in M} \dfrac{X'_{G'}}{BINS(V)} s'_i \cdot Y_{i,V}$ in the right hand size of the constraint (7) for W, and an increase of $\omega \cdot X'_{G'}$ in the left hand side. Since we have $\omega \cdot BINS(V) \geq \omega' \cdot BINS(V) \geq \sum_{i \in M} s'_i \cdot Y_{i,V}$ before the modification occurs (since constraint (7) holds for the solution before modification for the window V), we get that the total increase of the left hand side is no smaller than the total increase in the right hand side. There is an increase of $\sum_{i \in M} \dfrac{X'_{G'}}{BINS(V)} \cdot Y_{i,V}$ in the right hand size of the constraint (8) for W, and an increase of $k_t \cdot X'_{G'}$ in the left hand side. Since we have $k_t \cdot BINS(V) \geq k_{t'} \cdot BINS(V) \geq \sum_{i \in M} Y_{i,V}$, we get that the increase of the left hand side is no smaller than the increase in the right hand side. ∎

Now, we can temporarily delete the constraints of (7) and (8) that correspond to windows in $\mathcal{W} \setminus \mathcal{W}'$. We call the resulting linear program LP_{tmp}. We consider a basic solution of LP_{tmp} that is not worse than the solution we obtained above (which was created as a solution of LP_{tmp} too). Such a basic solution can be found in polynomial time. We denote this basic solution by $(\mathcal{X}, \mathcal{Y})$. This is clearly a basic solution to the original linear program as well.

In order to obtain a feasible packing, we will use the solution $(\mathcal{X}, \mathcal{Y})$. However, this solution may contain fractional components. We can show the following bound on the number of these components.

Lemma 18 *Consider the solution $(\mathcal{X}, \mathcal{Y})$. Let F_Y be the number of small items that are assigned to windows fractionally according to the solution, i.e., $F_Y = |\{i \in M, \text{ such that the vector } (\mathcal{Y}_{i,W})_{W \in \mathcal{W}}$ is fractional\}|. Let F_X be the number of fractional components of $\mathcal{X}$, i.e., the number of configurations assigned a non-integer number of copies in the solution. Then $F_Y + F_X \leq |H| + 2|\mathcal{W}'|$. If $M = \emptyset$, then $F_Y + F_X \leq |H|$.*

Proof. The linear program LP_{tmp} consists of $|H| + 2|\mathcal{W}'| + |M|$ inequality constraints, and hence in a basic solution (a property that we assume that $(\mathcal{X}, \mathcal{Y})$ satisfies) there are at most $|H| + 2|\mathcal{W}'| + |M|$ basic variables. For every $i \in M$, there is at least one window W such that $Y_{i,W}$ is a basic variable, and therefore there are at most $|H| + 2|\mathcal{W}'|$ additional fractional components in $(\mathcal{X}, \mathcal{Y})$. If $M = \emptyset$, then there are only $|H|$ constraints in the linear program, so there are at most $|H|$ basic variables. ∎

Rounding the solution. We apply several steps of rounding to obtain a feasible packing of the items into bins. Let C_{LP} be the cost obtained in the linear program by the vector $(\mathcal{X}, \mathcal{Y})$. By Lemma 16, this cost is at most $(1 + \varepsilon)^2 \text{OPT}'(I'')$.

For each $i \in M$ such that the vector $(\mathcal{Y}_{i,W})_{W \in \mathcal{W}}$ is fractional, i is packed in a dedicated bin. We can therefore assume that for every small item $i \in M$ to be packed, $(\mathcal{Y}_{i,W})_{W \in \mathcal{W}}$ is integral. Without loss of generality, we assume that it has one component equal to 1, and all other components are zero.

(If this is not the case, we can modify the vector without changing the feasibility of the solution, or the value of the objective function.)

Let $\hat{X}$ be the vector such that $\hat{X}_G = \lceil \mathcal{X}_G \rceil$ for all $G \in \mathcal{G}$. The number of bins allocated to generalized configuration G is $\hat{X}_G$.

We pack the items of $L' = L \setminus L_1$ first. We initialize bins according to generalized configurations, and assign large items into these bins according to the associated configurations (some slots may remain empty).

Lemma 19 *The cost of the additional bins, dedicated to small items for which $(\mathcal{Y}_{i,W})_{W \in \mathcal{W}}$ is fractional, and the cost of additional bins that are created as a result of replacing $\mathcal{X}$ by $\hat{X}$ is at most $f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'|)$. If $M = \emptyset$ then the additional cost is at most $f(\frac{1}{\varepsilon})|H|$.*

Proof. We calculate the cost of bins opened in addition to the cost implied by the solution $(\mathcal{X}, \mathcal{Y})$. At most one bin containing at most k_Δ items was opened for every fractional component of $\mathcal{X}_G$. At most one bin containing a single item was opened for every small item that was assigned fractionally to windows. The cost of a bin of the first type is at most $f(k_\Delta)$. The cost of every bin of the second type is $f(1) = 1 \leq f(\frac{1}{\varepsilon}) \leq f(k_\Delta)$. The total number of the two types of bins together is at most $|H| + 2|\mathcal{W}'|$ by Lemma 18. If $M = \emptyset$, then by definition, $k_\Delta = \frac{1}{\varepsilon}$, and the number of additional bins is at most $|H|$. ∎

Before moving on to the specific assignment of small items, we complete the packing of the original large items. Each large item of the rounded up instance is replaced by the corresponding item of I. The method of rounding implies that the space allocated to the rounded items is sufficient for the original items. Moreover, every item is replaced by at most one item, so the cost does not increase.

Each item of L_1 is packed into one dedicated bin.

Lemma 20 *The cost of the bins dedicated to the items of L_1 is at most $2\varepsilon^2 \text{OPT}(I)$.*

Proof. It suffices to show that $f(1)|L_1| \leq 2\varepsilon^2 \text{OPT}(I)$. To see this last claim note that $|L_1| \leq 2|L|\varepsilon^3$ and each item in L has size at least ε and therefore the number of bins used by $\text{OPT}(I)$ is at least $|L|\varepsilon$, where each of them costs at least $f(1)$. Therefore, $f(1)|L_1| = |L_1| \leq 2\varepsilon^2 \text{OPT}(I)$. ∎

Since the bound of the cost of the solution, before the rounding of fractional variables, and before taking the bins of items of $L_1 = L \setminus L'$ into account was $(1 + \varepsilon)^2 \text{OPT}'(I'')$, the current bound is $(1 + \varepsilon)^2 \text{OPT}'(I'') + f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'|) + 2\varepsilon^2 \text{OPT}(I)$. If $M = \emptyset$, then the bound is at most $(1 + \varepsilon)^2 \text{OPT}'(I'') + f(\frac{1}{\varepsilon}) \cdot |H| + 2\varepsilon^2 \text{OPT}(I)$, in which case the current solution implies a valid packing (together with the packing of S, whose cost is $F(S)$, see Definition 4), and thus the following bound on the final cost is established.

Lemma 21 *If $M = \emptyset$, then the algorithm returns a solution that costs at most $(1 + \varepsilon)^2 \text{OPT}'(I'') + f(\frac{1}{\varepsilon}) \cdot |H| + 2\varepsilon^2 \text{OPT}(I) + F(S)$.*

By the constraints (5), the allocation of the items of L' to slots reserved for such items is successful. At this time, we have removed some small items into new bins, and possibly increased the space allocated to other small items.

Packing the small items. We next consider the packing of the small items that are supposed to be packed (according to $\mathcal{Y}$) in bins with window W. Assume that there are $X(W)$ such bins (i.e., $X(W) = \sum\limits_{G=((C,p),W)} \hat{X}_G$). Denote by $S(W)$ the set of small items of M that according to $\mathcal{Y}$ are

supposed to be packed in bins with window W (for some of these items we will change this decision later). Then, by the feasibility of the linear program we conclude that $\sum_{i \in S(W)} s'_i \leq \omega \cdot X(W)$ and $|S(W)| \leq k_t \cdot X(W)$ for any $W = (\omega, t) \in \mathcal{W}'$.

We next show how to allocate almost all the items of $S(W)$ to the $X(W)$ bins with window $W = (\omega, t)$ such that the total size of items of $S(W)$ in each such bin will be at most $1 + \frac{\varepsilon\omega}{1+\varepsilon}$ and the total number of items of $S(W)$ in each such bin will be at most k_t.

To do so, we sort the items in $S(W)$ according to non-increasing size (assume the sorted list of item indices is $b_1 < b_2 < \ldots b_{|S(W)|}$). Then, allocate the items to the bins in a round-robin manner, so that bin j ($1 \leq j \leq X(W)$) receives items of indices $b_{j+q \cdot X(W)}$ for all integers $q \geq 0$ such that $j + q \cdot X(W) \leq |S(W)|$. We call the allocation of items for a given value of q a *round of allocations*. This process is applied for every window $W = (\omega, t) \in \mathcal{W}'$. If $\omega = \frac{s'_{min}}{1+\varepsilon}$ then there are no small items assigned to this window. We therefore assume that a window for which the process is applied satisfies $\omega \geq s'_{min}$. The resulting (possibly invalid) solution is called SOL_{start}.

Lemma 22 *In the solution SOL_{start}, the last bin of index $X(W)$ received at most an $\frac{1}{X(W)}$ fraction of the total size of the items, which is $\sum_{i=1}^{|S(W)|} s_{b_i}$.*

Proof. We artificially add at most $X(W) - 1$ items of size zero to the end of the list (these items are added just for the sake of the proof), and allocate them to the bins that previously did not receive an item in the last round of allocations, that is, bins $r, \ldots, X(W)$ such that bin $r - 1 < X(W)$ originally received the last item. If bin $X(W)$ received the last item then no items are added. Now the total size of small items remained the same, but every bin got exactly one item in each round. Since the last bin received the smallest item in each round, the claim follows. ∎

On the other hand, we can apply the following process, at every time $i < X(W)$, remove the first (largest) small item from bin i. As a result, the round-robin assignment now starts from bin $i+1$ and bin i becomes the bin that receives items last in every round, and thus by the previous proof, the total size of items assigned to it (after this removal) is at most $\frac{\sum_{i=1}^{|S(W)|} s_{b_i}}{X(W)}$ (since the total size of items does not increase in each step of removal).

We create an intermediate solution SOL_{inter}. In order to create this solution we apply the following process. The largest small item from each bin which received small items is removed (call them *the removed small items*). Each removed small item is indeed small and therefore its size is at most ε. We pack the removed small items in new bins, so that each bin contains $\frac{1}{\varepsilon}$ items. There may be at most one resulting bin with less than $\frac{1}{\varepsilon}$ items.

The solution SOL_{inter} is not necessarily valid because our definition of ω is larger than the available space for small items in such bin. If we temporarily relax the condition on the total size of items in a bin, we can compute its cost. Since the assignment of small items into bins is done using a round-robin method, the number of small items in a bin with a window (ω, p) is at most k_p.

Lemma 23 *The total cost of SOL_{inter} is at most the sum of $f(\frac{1}{\varepsilon})$ plus $(1 + \varepsilon)^2$ times the cost of the solution prior to the allocation of the small items into bins, that is, it is at most*

$$(1 + \varepsilon)^2 \left((1 + \varepsilon)^2 \text{OPT}'(I'') + f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'|) + 2\varepsilon^2 \text{OPT}(I) \right) + f(\frac{1}{\varepsilon}).$$

Proof. The first multiplicative factor of $1 + \varepsilon$ follows from Corollary 14. We next calculate the cost of the additional bins. We allocate a cost of $\varepsilon f(\frac{1}{\varepsilon})$ to each removed small item. Then, the total allocated cost covers that cost of all new bins except for at most one bin that has a cost of at most $f(\frac{1}{\varepsilon})$. Consider a removed item i and let a be the real number of items (including large items) that the bin from which i is removed, contains before the removal. Thus, the bin is charged with a cost of of at least $f(a)$ (the linear program may have charged it with $f(k_p)$ for some $k_p \leq a$, but the current charge for this bin in our estimation of the total cost is $(1 + \varepsilon)f(k_p) \geq f(a)$, by Corollary 14). As a result of removal of i, the real cost of the bin is no larger than $f(a - 1)$. We therefore next show that $\varepsilon f(\frac{1}{\varepsilon}) + f(a - 1) \leq (1 + \varepsilon)f(a)$. If $a \geq \frac{1}{\varepsilon}$, then using monotonicity, $f(\frac{1}{\varepsilon}) \leq f(a)$ and

$f(a - 1) \leq f(a)$ so the claim holds. Otherwise, we have $f(\frac{1}{\varepsilon}) = f(a) + \sum\limits_{j=a+1}^{\frac{1}{\varepsilon}} (f(j) - f(j - 1))$.

By concavity, we have for every $a + 1 \leq j \leq \frac{1}{\varepsilon}$, $f(a) - f(a - 1) \geq f(j) - f(j - 1)$. Therefore $f(\frac{1}{\varepsilon}) \leq f(a) + \frac{1}{\varepsilon}(f(a) - f(a - 1))$. Rewriting this gives the required claim. ∎

We note that the total size of small items assigned to such (original) bin is at most ω (as before removing the items we allocate the last bin a total size that is at most ω and after the removal of items each bin has total size which is at most the total size of the last bin before the removal).

Recall that the intermediate solution SOL_{inter} may be infeasible. We create the final solution SOL_{final} as follows. Consider a bin of the intermediate solution in which large items are packed according to configuration C, and small items have total size at most ω. We do not change the packing of large items. As for the small items, we remove them from the bin and start packing the small items again into this bin greedily in non-decreasing order of the item sizes, as long as the total size of items packed to the bin does not exceed 1. The first item that does not fit into the bin is called the *special item*. Additional items that do not fit are called the *excess items*.

We collect the special items from all bins, and we pack these items in separate bins, so that each such separate bin will contain $\frac{1}{\varepsilon}$ special items from different bins of SOL_{inter}, except perhaps for the last such bin.

By the definition of windows, the actual space in a bin with window (ω, t), that is free for the use of small items, is at least of size $\frac{\omega}{1+\varepsilon}$. After the removal of the packed items and the special item, we are left with the excess items. Let $\omega' \leq \omega$ be the space occupied by small items in the bin, before the removal of the special item and the excess items. Then the total size of excess items is at most $\omega' - \frac{\omega}{1+\varepsilon} \leq \omega' - \frac{\omega'}{1+\varepsilon} \leq \varepsilon \frac{\omega'}{1+\varepsilon} < \varepsilon$. Similar considerations can be applied to the cardinality of these items. Let ρ be the number of excess items. Since we insert the items into the bin sorted by a non-decreasing order of size, the largest items are the ones that become excess items. Thus, since there are at most k_t small items, the total size of the ρ largest items is at least $\frac{\rho}{k_t} \cdot \frac{\omega'}{1+\varepsilon}$, so $\varepsilon \frac{\omega'}{1+\varepsilon} \geq \frac{\rho}{k_t} \cdot \frac{\omega'}{1+\varepsilon}$. Thus for a window (ω, t), the number of excess items is at most εk_t.

The last rounding step is defined as follows. We can pack the unpacked (excess) items of every $\frac{1}{\varepsilon}$ bins of SOL_{inter} using one additional bin. Specifically, we sort the subsets of excess items according to a non-increasing order of the second component of the windows to which these items were originally assigned, we call it the *index of the subset*. Then, according to this order, we assign every consecutive $\frac{1}{\varepsilon}$ subsets to a bin. The last bin may contain a smaller number of subsets. This completes the scheme. We get our final solution SOL_{final}.

Bounding the cost of the resulting solution.

Lemma 24 *The cost of SOL_{final} is at most $(1 + 2\varepsilon)$ times the cost of SOL_{inter} plus $f(k_\Delta) + f(\frac{1}{\varepsilon})$.*

Proof. Similarly to the proof of Lemma 23, the bins which are used for packing the special items are feasible bins and they add ε times the cost of SOL_{inter} to the total cost of the packing plus $f(\frac{1}{\varepsilon})$. It remains to bound the additional cost of the bins which are used for the excess items.

We use $\kappa(i)$ to denote the index of the i-th subset. Let v denote the number of bins created, and u the number of subsets (we have $\frac{v-1}{\varepsilon} < u \leq \frac{v}{\varepsilon}$). The number of items in the i-th bin, for $i \geq 2$, is at most $\sum_{j=1}^{\frac{1}{\varepsilon}} \varepsilon k_{\kappa(\frac{i-1}{\varepsilon}+j)} \leq k_{\kappa(\frac{i-1}{\varepsilon})}$. The number of items in the first bin is at most $\Delta \leq k_\Delta$. The cost of the bins is therefore at most $f(\Delta) + \sum_{i=1}^{v-1} f\left(k_{\kappa(\frac{i}{\varepsilon})}\right)$. On the other hand, the cost of SOL_{inter} that is charged to the bin which was supposed to get the i-th subset of excess items is at least $f(k_{\kappa(i)})$ (since for a generalized configuration $((C,p),(\omega,t))$ we have $k_t \leq k_p$), thus the cost of SOL_{inter} is at least $\sum_{j=1}^{u} f(k_{\kappa(j)}) \geq \frac{1}{\varepsilon} \sum_{j=1}^{v-1} f\left(k_{\kappa(\frac{j}{\varepsilon})}\right)$. Thus the additional cost is at most ε times the cost of SOL_{inter} plus $f(\Delta)$. ∎

First assume that $M \neq \emptyset$. By Lemmas 23, 24, and by using $\varepsilon \leq \frac{1}{\varepsilon}$, the cost of SOL_{final} is at most

$$(1+2\varepsilon)\cdot\left((1+\varepsilon)^2 \cdot \left((1+\varepsilon)^2 \text{OPT}'(I'') + f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'|) + 2\varepsilon^2 \text{OPT}(I)\right) + f(\tfrac{1}{\varepsilon})\right) + f(k_\Delta) + f(\tfrac{1}{\varepsilon})$$

$$\leq (1+\varepsilon)^6 \text{OPT}'(I'') + 3f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'| + 1) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I).$$

SOL_{final} is a packing of $L \cup M$. In order to get the cost of the entire input, we need to add the cost of the packing of S, which was denoted by $F(S)$ (see Definition 4), which gives a total cost of at most

$$(1+\varepsilon)^6 \text{OPT}'(I'') + 3f(k_\Delta) \cdot (|H| + 2|\mathcal{W}'| + 1) + F(S) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I).$$

Using (4) this is at most

$$(1+\varepsilon)^6 \text{OPT}'(I'') + (1+\varepsilon)\left(\frac{3(|H| + 2|\mathcal{W}'| + 1)}{h(\varepsilon)} + 1\right) F(S) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I) \qquad (11)$$

Since every element of $\mathcal{W}'$ is the main window of an extended configuration, we have $|\mathcal{W}'| \leq |\mathcal{C}|$. Every configuration has at most $\frac{1}{\varepsilon}$ items, of $|H|$ sizes, and an extended configuration has a second component which is an integer in $\{1, 2, \ldots, \Delta\}$, where $\Delta \leq \ell$, so we have $|\mathcal{C}| \leq \ell \cdot (|H| + 1)^{\frac{1}{\varepsilon}} \leq \ell \cdot (\frac{1}{\varepsilon^3} + 1)^{1/\varepsilon}$. In addition, $1 \leq |H| \leq |\mathcal{W}'| \leq \ell \cdot (\frac{1}{\varepsilon^3} + 1)^{1/\varepsilon}$. This gives $3(|H| + 2|\mathcal{W}'| + 1) \leq 12\ell \cdot (\frac{1}{\varepsilon^3} + 1)^{1/\varepsilon} = 2\varepsilon \cdot h(\varepsilon)$, using the definition of $h(\varepsilon)$ in (1). Thus the value of (11) is at most

$$(1+\varepsilon)^6 \text{OPT}'(I'') + (1+\varepsilon)(1+2\varepsilon)F(S) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I)$$

$$\leq \quad (1+\varepsilon)^6 (\text{OPT}'(I'') + F(S)) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I)$$

By Definition 4, and by Lemma 12 and using $\varepsilon \leq \frac{1}{3}$ again, this is at most

$$(1+\varepsilon)^6 (\text{OPT}'(I')) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2 \text{OPT}(I) \qquad (12)$$

$$\leq \quad (1+\varepsilon)^6\left((1+\varepsilon)\mathrm{OPT}(I) + (3h(\varepsilon)+3)f(\tfrac{1}{\varepsilon})\right) + 3f(\tfrac{1}{\varepsilon}) + 6\varepsilon^2\mathrm{OPT}(I) \qquad (13)$$

$$\leq \quad (1+22\varepsilon)\mathrm{OPT}(I) + f(\tfrac{1}{\varepsilon})(17h(\varepsilon)+20)$$

To complete the proof, starting from equation (12), we use only properties which hold even if $M = \emptyset$, so that we can reduce the remaining cases to the current case. We next use the property that by concavity of f we have $f(z) \leq z \cdot f(1) = z$ for any $z \geq 1$. Therefore, $f(\tfrac{1}{\varepsilon}) \leq \tfrac{1}{\varepsilon}$, and using the bound on ℓ (Equation (3)), we get that the last amount is at most

$$(1+22\varepsilon)\mathrm{OPT}(I) + 17\frac{h(\varepsilon)}{\varepsilon} + \frac{20}{\varepsilon} \leq (1+22\varepsilon)\mathrm{OPT}(I) + 102\frac{\ell(\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}}}{\varepsilon^2} + \frac{20}{\varepsilon}$$

$$\leq \quad (1+22\varepsilon)\mathrm{OPT}(I) + \frac{20}{\varepsilon} + \frac{102((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})}{\varepsilon^2}\left(\frac{1}{\varepsilon} + 1 + 2\log_{1+\varepsilon}\mathrm{OPT}(I)\right) \qquad (14)$$

We note that the last bound can be written as $(1 + 22\varepsilon) \cdot \mathrm{OPT} + \phi(\varepsilon) \cdot \log_2 \mathrm{OPT} + \psi(\varepsilon)$ where ϕ and ψ are some (exponential) functions of $\tfrac{1}{\varepsilon}$. To show that the resulting scheme is an AFPTAS it suffices to argue that $\phi(\varepsilon) \cdot \log_2 \mathrm{OPT} \leq \varepsilon\mathrm{OPT} + \left(\frac{\phi(\varepsilon)}{\varepsilon}\right)^2 + 4$. To see this last inequality note that if $\log_2 \mathrm{OPT} \leq \frac{\phi(\varepsilon)}{\varepsilon^2}$ the claim clearly holds. Otherwise, if $\mathrm{OPT} \leq 16$ and $\phi(\varepsilon) < 1$ the claim holds. It remains to consider the case where $\mathrm{OPT} > 16$ or $\phi(\varepsilon) \geq 1$. Note that if $\phi(\varepsilon) \geq 1$, then since we assume $\log_2 \mathrm{OPT} > \frac{\phi(\varepsilon)}{\varepsilon^2} \geq \frac{1}{\varepsilon^2} \geq 9$ and therefore $\mathrm{OPT} > 16$ holds as well. Thus, we only need to consider the case $\mathrm{OPT} > 16$. For any $x > 16$ we have $\sqrt{x} \geq \log_2 x$ and by $\mathrm{OPT} \geq 16$, we get $\sqrt{\mathrm{OPT}} \geq \log_2 \mathrm{OPT} \geq \frac{\phi(\varepsilon)}{\varepsilon^2}$. Therefore, $\varepsilon\mathrm{OPT} = \varepsilon\sqrt{\mathrm{OPT}}\sqrt{\mathrm{OPT}} \geq \frac{\phi(\varepsilon)}{\varepsilon}\sqrt{\mathrm{OPT}} \geq \frac{\phi(\varepsilon)}{\varepsilon}\log_2 \mathrm{OPT} \geq \phi(\varepsilon)\log_2 \mathrm{OPT}$ and the claim follows.

Thus, letting $\phi(\varepsilon) = \frac{204}{\varepsilon^2} \cdot (\log_{1+\varepsilon} 2) \cdot ((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})$ the amount in (14) is equal to

$$(1+22\varepsilon)\mathrm{OPT}(I) + \frac{20}{\varepsilon} + \frac{102((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})}{\varepsilon^2}\left(\frac{1}{\varepsilon} + 1\right) + \phi(\varepsilon)\log_2\mathrm{OPT}(I)$$

$$\leq \quad (1+22\varepsilon)\mathrm{OPT}(I) + \frac{20}{\varepsilon} + \frac{102((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})}{\varepsilon^2}\left(\frac{1}{\varepsilon} + 1\right) + \varepsilon\mathrm{OPT}(I) + \left(\frac{\phi(\varepsilon)}{\varepsilon}\right)^2 + 4$$

Since $(1+\varepsilon)^{\tfrac{1}{\varepsilon}} \geq 2$, we have $\log_{1+\varepsilon} 2 \leq \tfrac{1}{\varepsilon}$, so $\phi(\varepsilon) \leq \frac{204}{\varepsilon^3}((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})$, so the upper bound on the cost is at most $(1+23\varepsilon)\mathrm{OPT}(I) + \upsilon(\varepsilon)$, where

$$\upsilon(\varepsilon) = \frac{20}{\varepsilon} + \frac{102((\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}})}{\varepsilon^2}\left(\frac{1}{\varepsilon} + 1\right) + \left(\frac{204(\tfrac{1}{\varepsilon^3}+1)^{\tfrac{1}{\varepsilon}}}{\varepsilon^4}\right)^2 + 4 \ .$$

If $M = \emptyset$, we get $k_\Delta = \tfrac{1}{\varepsilon}$ so $f(k_\Delta) \leq \tfrac{1}{\varepsilon}$. Assume that $I'' = M \cup L' = \emptyset$. In this case we showed in Corollary 13 that the cost of the returned solution is at most $(3h(\varepsilon)+3) \cdot f(\tfrac{1}{\varepsilon})$. The cost of the solution is at most the amount in (13), so the previous proof covers this case and thus we get an AFPTAS in this case as well.

It remains to consider the case $M = \emptyset$ (and $L' \neq \emptyset$). In this case there are no small items to be packed by the linear program, and we saw in Lemma 21 that the the cost is at most $(1+\varepsilon)^2\mathrm{OPT}'(I'') + f(\tfrac{1}{\varepsilon}) \cdot |H| + 2\varepsilon^2\mathrm{OPT}(I) + F(S)$.

Thus using $|H| \leq \frac{1}{\varepsilon^3}$, we conclude that the cost of the returned solution is at most

$$(1 + \varepsilon)^2(\mathrm{OPT}'(I'') + F(S)) + f(\frac{1}{\varepsilon}) \cdot |H| + 2\varepsilon^2\mathrm{OPT}(I)$$

$$\leq \quad (1 + \varepsilon)^2\mathrm{OPT}'(I') + \frac{1}{\varepsilon^4} + 2\varepsilon^2\mathrm{OPT}(I)$$

The last expression exceeds the value of (12) by at most $\frac{1}{\varepsilon^4}$, and therefore, we obtain an AFPTAS in this last case as well, and we have established the correctness of Theorem 11.

To conclude, we summarize the core of the AFPTAS presented in this section.

A sketch of the main steps of the AFPTAS

1. Partition the input into $\cup_{i=1}^m L_i \cup M \cup S$. Pack each item of L_1 into a dedicated bin. Use rounded values of other items.

2. Pack S using FNFI; repack split items in dedicated bins.

3. Determine a set of breakpoints for f to get a staircase approximation.

4. Use the column generation technique to approximately solve the dual linear program of the configuration linear program, repeatedly calling the FPTAS of [8]. The solution is given in a compact way.

5. Modify the solution so that all the windows which are not main windows of any configuration are not used. This is done by replacing the windows by larger windows.

6. Convert the solution to a basic solution to the linear program LP_{tmp}.

7. Round the solution to get an integer number of instances of each generalized configuration. Every small item which was not assigned integrally into a window type is packed into a dedicated bin.

8. Pack the rounded large items to bins according to the configurations of the solution.

9. Assign the small items. For each type of window, and a set of items assigned to this type of window by the solution, consider all bins whose configuration has this window. Assign the small items in a round-robin manner.

10. Remove one item from each bin that has small items, and pack those items separately ($\frac{1}{\varepsilon}$ items per bin).

11. Remove additional small items until all bins are packed up to a total size of at most 1, and pack those small items separately, so that the items removed from every $\frac{1}{\varepsilon}$ bins are packed into a common bin.

12. Revert to the original values of the large items.

References

[1] S. Anily, J. Bramel, and D. Simchi-Levi. Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operations Research*, 42(2):287–298, 1994.

[2] B. S. Baker and E. G. Coffman, Jr. A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM Journal on Algebraic and Discrete Methods*, 2(2):147–152, 1981.

[3] N. Bansal, A. Caprara, and M. Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009.

[4] N. Bansal, J.R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31:31–49, 2006.

[5] W. W. Bein, J. R. Correa, and X. Han. A fast asymptotic approximation scheme for bin packing with rejection. *Theoretical Computer Science*, 393(1-3):14–22, 2008.

[6] J. Bramel, W. T. Rhee, and D. Simchi-Levi. Average-case analysis of the bin packing problem with general cost structures. *Naval Research Logistics*, 44(7):673–686, 1998.

[7] A. Caprara, H. Kellerer, and U. Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 92:58–69, 2003.

[8] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123:333–345, 2000.

[9] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.

[10] E. G. Coffman Jr. and J. Csirik. Performance guarantees for one-dimensional bin packing. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 32. Chapman & Hall/Crc, 2007. 18 pages.

[11] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, chapter 7, pages 147–177. Springer, 1998.

[12] 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.

[13] G. Dósa and Y. He. Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, 204(5):795–815, 2006.

[14] L. Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.

[15] L. Epstein. Bin packing with rejection revisited. *Algorithmica*, 56(4):505–528, 2010.

[16] L. Epstein and A. Levin. AFPTAS results for common variants of bin packing: A new method to handle the small items. Manuscript, 2007.

[17] L. Epstein and A. Levin. An APTAS for generalized cost variable-sized bin packing. *SIAM Journal on Computing*, 38(1):411–428, 2008.

[18] K. Jansen and R. van Stee. On strip packing with rotations. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC2005)*, pages 755–761, 2005.

[19] D. S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, MIT, Cambridge, MA, 1973.

[20] D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.

[21] D. S. Johnson, A. Demers, J. D. Ullman, Michael R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.

[22] 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.

[23] K. L. Krause, V. Y. Shen, and H. D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM*, 22(4):522–550, 1975.

[24] C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.

[25] C.-L. Li and Z.-L. Chen. Bin-packing problem with concave costs of bin utilization. *Naval Research Logistics*, 53(4):298–308, 2006.

[26] F. D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149–161, 1987.

[27] S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

[28] H. Shachnai and O. Yehezkely. Fast asymptotic FPTAS for packing fragmentable items with costs. In *Proc. of the 16th International Symposium on Fundamentals of Computation Theory, (FCT2007)*, pages 482–493, 2007.

[29] J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.