

Parameterized Approximation Scheme for the Multiple Knapsack Problem *

Klaus Jansen[†]

Abstract

The multiple knapsack problem (MKP) is a well-known generalization of the classical knapsack problem. We are given a set A of n items and set B of m bins (knapsacks) such that each item $a \in A$ has a size $size(a)$ and a profit value $profit(a)$, and each bin $b \in B$ has a capacity $c(b)$. The goal is to find a subset $U \subset A$ of maximum total profit such that U can be packed into B without exceeding the capacities. The decision version of MKP is strongly NP-complete, since it is a generalization of the classical knapsack and bin packing problem. Furthermore, MKP does not admit an FPTAS even if the number m of bins is two. Kellerer gave a PTAS for MKP with identical capacities and Chekuri and Khanna presented a PTAS for MKP with general capacities with running time $n^{O(\log(1/\epsilon)/\epsilon^8)}$. In this paper we propose an EPTAS with parameterized running time $2^{O(\log(1/\epsilon)/\epsilon^9)} \cdot poly(n) + O(m)$ for MKP. This solves also an open question by Chekuri and Khanna.

1 Introduction.

The knapsack problem is one of the fundamental and best known problems in combinatorial optimization. The multiple knapsack problem (MKP) is a generalization of the classical knapsack problem. We are given a set A of n items and set B of m bins (knapsacks) such that each item $a \in A$ has a size $size(a)$ and a profit value $profit(a)$, and each bin $b \in B$ has a capacity $c(b)$. For a subset $U \subset A$, let $size(U) = \sum_{a \in U} size(a)$ its size and $profit(U) = \sum_{a \in U} profit(a)$ its profit. The goal is to find a subset $U \subset A$ of maximum total profit $profit(U)$ such that U can be packed into B without exceeding the capacities. Let $OPT(A, B)$ be the maximum total profit among all feasible subsets $U \subset A$. The decision version of MKP which asks whether a set $U \subset A$ with total profit $profit(U) \geq p$ exists for a given value p is also a generalization of the classical bin packing problem and, therefore, strongly NP-complete.

MKP and its restrictions, e.g. with identical bins $c(b_i) = c$ for $i = 1, \dots, m$ or the variant where the

profits are equal to the sizes $profit(a) = size(a)$ for all items $a \in A$, capture several fundamental combinatorial optimization problems and have many practical applications in computer science, operations research, and related disciplines. The books on knapsack problems by Martello and Toth [15] and by Kellerer, Pferschy, and Pisinger [12] both have a full chapter devoted to MKP. An interesting application arises in scheduling jobs on identical processors with reservations or fixed jobs [5, 18]. In this case either high-priority jobs are preassigned to machines or machines are non-available due to maintenance during fixed intervals. A time interval where a processor is available can be seen as a bin and the jobs can be interpreted as items with different sizes. Finding a subset of jobs U with maximum total profit that can be executed by a deadline d is equivalent to the multiple knapsack problem.

A maximization problem X admits a polynomial-time approximation scheme (PTAS), if there is a family of algorithms $\{A_\epsilon \mid \epsilon > 0\}$ such that for any $\epsilon > 0$ and any instance I of X , A_ϵ produces a $(1 - \epsilon)$ -approximate solution in time $|I|^{f(1/\epsilon)}$ for some function f . If ϵ is very small then the exponent of the polynomial $|I|^{f(1/\epsilon)}$ can be very large. Two important restricted classes of approximation schemes were defined to avoid this dependence. An efficient polynomial-time approximation scheme (EPTAS) is a PTAS with running time of the form $f(1/\epsilon)|I|^{O(1)}$, while a fully time polynomial time approximation scheme (FPTAS) runs in time $(1/\epsilon)^{O(1)}|I|^{O(1)}$.

Results. In contrast to the classical knapsack problem, the multiple knapsack problem (MKP) does not admit an FPTAS even if the number of bins is two (unless $P = NP$) [4]. Kellerer [11] gave a polynomial time approximation scheme for MKP with identical capacities. This result has been generalized by Chekuri and Khanna [4] who gave a PTAS for MKP with general capacities. They also gave a $2 + \epsilon$ approximation algorithm. It is based on a natural greedy strategy: pack bins one at a time, by applying an FPTAS for the classical problem [12, 13] on the remaining items. *Greedy*(ϵ) refers to this algorithm with ϵ parameterizing the error tolerance used in the knapsack FPTAS. It produces a solution U with total profit $profit(U) \geq (1/2 - \epsilon/2)OPT(A, B)$. The running time

*Supported in part by EU project, Algorithmic Principles for Building Efficient Overlay Computers, contract number 015964.

[†]Institut für Informatik, Universität zu Kiel, Christian-Albrechts-Platz 4, 24098 Kiel, Germany. E-mail: kj@informatik.uni-kiel.de

of the PTAS by Chekuri and Khanna is $n^{O(\log(1/\epsilon)/\epsilon^8)}$ [3]. Chekuri and Khanna posed as an interesting problem the question of whether there is a PTAS for MKP with an improved running time. An FPTAS is ruled out even for the case with two identical bins. On the other hand, they mention that a PTAS with a running time of the form $f(1/\epsilon)poly(n)$ might be achievable. Such an algorithm is not known even for instances in which all bins have the same capacity. For an error of 20 percent the running time of the PTAS by Chekuri and Khanna is very huge (mentioned by Downey [7]). For a survey about parameterized complexity and approximation algorithms we refer to a paper by Marx [16]. Fellows [8] mentioned it as a significant open problem whether MKP admits a fixed parameter tractable (FPT) algorithm or is $W[1]$ -hard. In this paper we prove the following result:

THEOREM 1.1. *There is an efficient polynomial-time approximation scheme (EPTAS) for the multiple knapsack problem with parameterized running time $2^{O(\log(1/\epsilon)/\epsilon^9)} \cdot poly(n) + O(m)$.*

Notice that it is sufficient to consider only the n largest bins for the case $m \geq n$. In this case, any solution that uses also other bins can be directly transformed into another solution that uses only the n bins with largest capacities. We can find the n largest bins in an unsorted sequence of m bins in $O(m)$ time [2, 6]. Therefore, we may assume in the following that $m \leq n$. Interestingly MKP with many bins of the same capacity is easier and faster to approximate. For the case where we have at least $\Omega(1/\epsilon^3)$ bins for each capacity c_i and $i = 1, \dots, t$, we propose an approximation algorithm that finds a solution with profit at least $(1 - \epsilon)OPT(A, B)$ and runs in time polynomial in n , t and $1/\epsilon$. This implies as a special case also a more efficient approximation scheme for MKP with identical bins. For the case with at most $\gamma \leq O(1/\epsilon^3)$ identical bins we obtain an approximation algorithm with running time $2^{O(\log(1/\epsilon)\gamma/\epsilon)}n = 2^{O(\log(1/\epsilon)/\epsilon^4)}n$. Furthermore, if $\gamma > \Omega(1/\epsilon^3)$ the running time of our approximation algorithm for identical bins is polynomial in n and $1/\epsilon$.

Techniques. In contrast to previous approaches we use several new and different techniques. For MKP with many bins for each capacity value, we propose an interesting linear program relaxation of MKP that is a variant of the configuration LP for fractional bin or strip packing. The main idea is to select fractional pieces of the items and to distribute them among different bins. The linear program relaxation has an exponential number of variables and some negative covering constraints that can be solved approximately via techniques for the

max-min resource sharing problem [9]. Interestingly, the LP solution with fractional selected items can be interpreted as rectangles to be placed into rectangular blocks of different widths and large heights. Then we set up another linear program to modify the distribution of the selected fractional items among the blocks. To do this we use some techniques by Kenyon and Remila [10] for 2D strip packing (i.e. the rounding of the wide rectangles via a stack and counting the area for the narrow rectangles) and by Lenstra, Shmoys and Tardos [14] for scheduling on unrelated machines (i.e. rounding the LP solution for the scheduling problem). The obtained solution gives us a unique assignment of almost all selected fractional items to blocks.

Since we have selected in general only fractional items, in a second phase we have to produce a solution with completely selected items. This can be done via fractional knapsack instances for the corresponding selected narrow and wide rectangles. Finally, we show that the selected items can be packed into the rectangular blocks or corresponding bins (via 2D strip packing) and that the profit of the solution (via the shifting technique) is close to the optimum profit. In the general case we modify the structure of the bins to obtain many bins for almost each capacity value. This is done via an interesting rounding step (linear rounding of all bins except a constant number of large ones). We end up with a constant number of large capacity values where the number of bins is small. For all other capacity values we obtain many bins and this makes it possible to use a modification of the techniques above. In the next step, we modify the structure of the high profit items to reduce the number of such items. Then we guess a constant number of high profit items to be placed into the constant number of large bins. For each such guess, we solve a modified linear program relaxation approximately and round the generated solution as described above.

Organization of the Paper. In Section 2 we consider the case where we have instances with many bins of the same capacity. Here we study in detail a linear program relaxation of MKP, rounding of the fractional LP solution, selecting items via fractional knapsack problems and packing the items into the bins via 2D strip packing. Then in Section 3 we analyze instances with a constant number of bins. For this problem we show how to reduce the instance to a constant number of items with high profit and to select the other items via a single knapsack algorithm. Finally, we study the general case in Section 4. Here we first modify the bin structure. Then we guess a subset of items with high profit to be placed into few bins and use a modified linear program relaxation.

2 Instances with Similar Capacities.

Let c_1, \dots, c_t be the different capacities in an instance of the multiple knapsack problem. Let us consider in this section the case where for each capacity c_ℓ there are $m_\ell \geq 1/\delta^3$ bins of capacity c_ℓ . In this case, $m = \sum_{\ell=1}^t m_\ell \geq t/\delta^3$, or equivalently $t \leq \delta^3 m$. If $m \geq n$ then we can remove again the smaller bins until we have at most n bins (the group with the lowest capacities may have less than $1/\delta^3$ bins). Therefore, we may assume that $t \leq \delta^3 m \leq \delta^3 n$.

2.1 Linear Program Relaxation. Let $A = \{a_1, \dots, a_n\}$ be the set of items. For each c_ℓ we consider configurations or subsets $A' \subset A$ of the items with total size $\sum_{a \in A'} \text{size}(a) \leq c_\ell$ (such a subset can be packed into a bin of capacity c_ℓ). Let $C_1^{(\ell)}, \dots, C_{H_\ell}^{(\ell)}$ be the set of configurations for group ℓ (of bins of capacity c_ℓ). The main idea is to use a fractional $x_i \in [0, 1]$ piece of each item a_i and to allow this piece to be distributed as smaller fractional pieces among the groups $\ell = 1, \dots, t$. In the LP below the variable $y_j^{(\ell)}$ denotes the length of the configuration $C_j^{(\ell)}$ in the solution. The linear program LP has now the following form:

$$\begin{aligned} \max \quad & \sum_{i=1}^n \text{profit}(a_i) x_i \\ \sum_{\ell=1}^t \sum_{j: a_i \in C_j^{(\ell)}} y_j^{(\ell)} &= x_i \quad \text{for } i = 1, \dots, n, \\ \sum_{j=1}^{H_\ell} y_j^{(\ell)} &\leq m_\ell \quad \text{for } \ell = 1, \dots, t, \\ y_j^{(\ell)} &\geq 0 \quad \text{for } j = 1, \dots, H_\ell \\ &\quad \text{and } \ell = 1, \dots, t, \\ x_i &\in [0, 1] \quad \text{for } i = 1, \dots, n. \end{aligned}$$

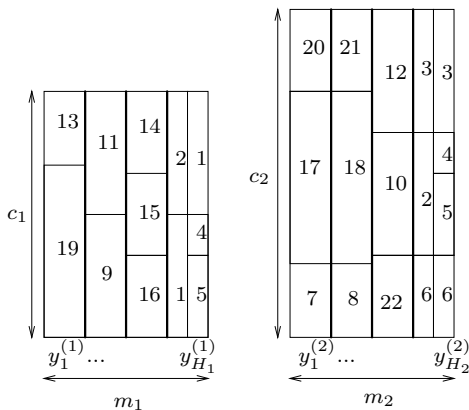


Figure 1: A feasible LP solution.

The value $\sum_{j: a_i \in C_j^{(\ell)}} y_j^{(\ell)}$ is exactly the part of the fractional piece a_i in group ℓ . Clearly, the sum of these

parts should be equal to x_i . For an example we refer to Figure 1. Here we have two bin capacities c_1, c_2 with $m_1 = m_2 = 4$ bins and a set of items $A = \{1, \dots, 22\}$. A feasible solution of the LP (indicated in Figure 1) has three configurations $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ with length 1 and two configurations $C_4^{(j)} = C_5^{(j)}$ with length $1/2$ for each capacity $c_j, j \in \{1, 2\}$.

LEMMA 2.1. *The linear program above is a relaxation of the multiple knapsack problem for instance (A, B) , i.e. the objective value of the LP satisfies: $OPT(LP) \geq OPT(A, B)$.*

Proof. Let us order the bins in B according to their capacities $c_1 < \dots < c_t$ where the bins with indices $\sum_{j=1}^{\ell-1} m_j + 1, \dots, \sum_{j=1}^{\ell} m_j$ have capacity c_ℓ for $\ell = 1, \dots, t$. In total we have m_ℓ bins of capacity c_ℓ . An optimum solution of MKP is a subset $A' \subset A$ and a distribution of A' among the bins in B ; the bins partition A' into subsets X_1, \dots, X_m where $m = |B|$ and bin b_i contains item set X_i . Each subset X_i with index $i \in \{\sum_{j=1}^{\ell-1} m_j + 1, \dots, \sum_{j=1}^{\ell} m_j\}$ corresponds to a configuration for group ℓ . Set the variable $y_j^{(\ell)} = 1$ (where the corresponding configuration $C_j^{(\ell)} = X_i$ for one of the indices above) and the others to 0. Then we have $\sum_{j=1}^{H_\ell} y_j^{(\ell)} \leq m_\ell$. On the other hand set variable $x_i = 1$ if the item $a_i \in A'$ and $x_i = 0$ otherwise. Since for each item $a_i \in A'$ there is a configuration $C_j^{(\ell)}$ (that contains a_i) with length $y_j^{(\ell)} = 1$ and $x_i = 1$, the first n equalities are satisfied directly. Furthermore, the objective value of this solution is equal to the profit $\sum_{a_i \in A'} \text{profit}(a_i)$. Since the LP allows also to take fractional pieces $x_i \in [0, 1]$ of items and to distribute them as pieces among the ℓ groups, we obtain $OPT(LP) \geq OPT(A, B)$. \square

In the full paper we show how to solve the LP as a max-min resource sharing problem and to obtain an approximate solution (\tilde{x}, \tilde{y}) where the sum of the configurations $\sum_{j=1}^{H_\ell} \tilde{y}_j^{(\ell)}$ is bounded by the number of bins m_ℓ times $(1 + 2\alpha)$ and whose objective value is at least $(1 - 3\alpha)$ times the optimum value $OPT(LP)$ of the LP or the optimum profit $OPT(A, B)$ of the multiple knapsack problem (where $\alpha = O(\epsilon)$).

2.2 Rounding the LP Solution. Now we study in more detail the splitting of the items into different bin groups. Let a_i be an item with $\tilde{x}_i > 0$. Define $z_i^{(\ell)} = \sum_{j: a_i \in C_j^{(\ell)}} \tilde{y}_j^{(\ell)}$ as the piece of item a_i assigned to group $\ell \in \{1, \dots, t\}$. For each group ℓ of bins with capacity c_ℓ , we think of all large pieces (with $\text{size}(a_i) \geq \delta c_\ell$) as rectangles of the form $(\text{size}(a_i), z_i^{(\ell)})$ with width $\text{size}(a_i)$ and height $z_i^{(\ell)}$.

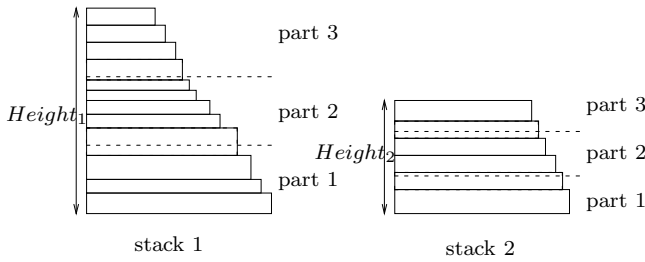


Figure 2: Building of 2 stacks.

Then we stack all these rectangles ordered by their widths. Let $Height_\ell$ be the height of stack ℓ . Each stack is divided into $1/\delta^2$ parts of height $\delta^2 Height_\ell$. We use the stacks and the corresponding splitting into parts for the rounding afterwards. The goal of the rounding is to obtain stacks with a similar shape where each item $a_i \in A$ lies as a rectangle piece in at most one stack. For an example we refer to Figure 2. Here we have two stacks with heights $Height_1$ and $Height_2$ that are both divided into three parts. Let K_ℓ be the set of pieces that lie in at least two parts of the stack ℓ . Notice that $|K_\ell| \leq 1/\delta^2$. Let us remove for now the items corresponding to the sets K_ℓ . By using a shifting technique afterwards we later may reinsert them back if their profit was large. From now on we suppose that there is at most one piece of each item on each stack. Let $S_{\ell,j}$ be the set of items a_i that have a piece $(size(a_i), z_i^{(\ell)})$ in part j of the ℓ .th stack. Furthermore, let $L_{wide}^{(\ell)}$ be the set of rectangles on stack ℓ . For the small pieces we simply compute the total area of the pieces allocated to group $\ell \in \{1, \dots, t\}$: $Area(\ell) = \sum_{i: z_i^{(\ell)} > 0, size(a_i) < \delta c_\ell} z_i^{(\ell)} size(a_i)$. For notation let $S_{\ell,0}$ be the set of items a_i with a small piece in the sum above and let $L_{narrow}^{(\ell)}$ be the corresponding set of narrow rectangles. Now we set up a linear program to round over the groups:

$$\begin{aligned} \sum_{a_i \in S_{\ell,j}} x_i^{(\ell)} &\leq \delta^2 Height_\ell && \text{for } \ell = 1, \dots, t \\ &&& \text{and } j = 1, \dots, 1/\delta^2, \\ \sum_{a_i \in S_{\ell,0}} x_i^{(\ell)} size(a_i) &\leq Area(\ell) && \text{for } \ell = 1, \dots, t, \\ \sum_{\ell=1}^t x_i^{(\ell)} &= \tilde{x}_i && \text{for } i = 1, \dots, n, \\ x_i^{(\ell)} &\geq 0 && \text{for } i = 1, \dots, n \\ &&& \text{and } \ell = 1, \dots, t. \end{aligned}$$

Notice that this linear program has a feasible solution $(\hat{z}_i^{(\ell)})$ (where $\hat{z}_i^{(\ell)} = z_i^{(\ell)}$ for $a_i \notin K_\ell$ and $\hat{z}_i^{(\ell)} = 0$ otherwise).

LEMMA 2.2. *The solution $(\hat{z}_i^{(\ell)})$ of the linear program above can be rounded into another solution $(\bar{z}_i^{(\ell)})$, where*

each set $L_\ell = \{a_i | \bar{z}_i^{(\ell)} \in (0, \tilde{x}_i)\}$ has at most $1/\delta^2 + 1$ items for $\ell = 1, \dots, t$.

Proof. The linear program can be transformed into a scheduling problem with jobs on unrelated machines. For this transformation we use $\bar{x}_i^{(\ell)} = x_i^{(\ell)}/\tilde{x}_i$, divide the inequalities by $\delta^2 Height_\ell$ and $Area(\ell)$, and obtain an equivalent linear program with right hand sides 1. The number M of machines in the formulation is $t(1/\delta^2 + 1)$, the number N of jobs is equal to the number n of items. One can round the solution into another feasible one $(\check{z}_i^{(\ell)})$ that has only few fractional variables $\check{z}_i^{(\ell)} \in (0, 1)$ (i.e. one fractional variable per machine); see Lenstra, Shmoys and Tardos [14] and Plotkin, Shmoys and Tardos [17]. By using $\bar{z}_i^{(\ell)} = \check{z}_i^{(\ell)} \tilde{x}_i$, we obtain the bounds above. \square

The running time of the rounding procedure can be bounded by $O(NM^2) = O(nt^2/\delta^4) = poly(n, 1/\epsilon)$ (using $t \leq \delta^3 n$ and $\delta = O(\epsilon)$).

2.3 Selecting the Items. After the rounding we have a unique assignment of items to groups (i.e. each item $a_i \in A \setminus \bigcup_\ell (K_\ell \cup L_\ell)$ with value $\tilde{x}_i > 0$ is now assigned to exactly one group ℓ and one part j). Let $\bar{S}_{\ell,j}$ be the set of items a_i with a piece in group ℓ and part j (i.e. with $\bar{z}_i^{(\ell)} = \tilde{x}_i$). Notice that the values $\tilde{x}_i \in (0, 1]$ are in general fractional. That means that our linear program selects pieces of the items. But we can select complete (non-fractional) items from our instance with near optimum profit that can be packed into the bins. This can be done by solving classical fractional knapsack problems for each group $\ell = 1, \dots, t$ and each part $j = 0, \dots, 1/\delta^2$. Simply take as instance the items $a_i \in \bar{S}_{\ell,j}$. For $j \geq 1$ take as size of an item the value 1 and as profit the original profit $profit(a_i)$. The capacity of the knapsack is equal to the height value $\delta^2 Height_\ell$. The advantage is that the structure of the selected items by the fractional knapsack problem is similar to the selected fractional items by the LP (see also the analysis in the next subsection). For group $j = 0$ take as size of an item $a_i \in \bar{S}_{\ell,0}$ the area $size(a_i) \cdot 1 = size(a_i)$ and as profit the original profit $profit(a_i)$. The capacity of the knapsack here is equal to the area value $Area(\ell)$. We solve now all fractional knapsack instances and obtain an overall solution $A' \subset A$ with at most one fractional item per group ℓ and part j . Let M_ℓ be the set of fractional items selected above in group ℓ . Again $|M_\ell| \leq (1/\delta^2 + 1)$. All other items $\bar{A} = A' \setminus \bigcup_\ell M_\ell$ are now selected completely. Let \bar{A}_ℓ be the selected items in group ℓ and $\bar{L}_{wide}^{(\ell)}, \bar{L}_{narrow}^{(\ell)}$ be the corresponding sets of wide and narrow rectangles with height 1. Since we have generated a solution of the

LP with value $(1 - 3\alpha)OPT(LP)$ and we do not lose any profit within the rounding and selection process, we obtain the following result:

LEMMA 2.3. *Let $X_\ell = \bar{A}_\ell \cup (K_\ell \cup L_\ell \cup M_\ell)$ for each $\ell = 1, \dots, t$. Then, $profit(\bigcup_\ell X_\ell) \geq (1 - 3\alpha)OPT(LP)$.*

The running time to solve all fractional knapsack problems can be bounded by $O(n)$. Here we use the fact that we have a unique distribution of at most n items and that there is a linear time $O(N)$ algorithm for the fractional knapsack problem with N items [1].

2.4 Strip Packing. It remains to show that the generated solution can be packed into the bins. To prove this we show that $\bar{A}_\ell \cup (K_\ell \cup L_\ell \cup M_\ell)$ can be packed into $(1 + O(\delta))m_\ell$ bins of capacity c_ℓ . This implies (using the shifting technique described below) that most of the items (with near optimum profit) can be packed also into m_ℓ bins. Using the property $\sum \tilde{y}_j^{(\ell)} \leq m_\ell(1 + 2\alpha)$ we know that the minimum fractional strip packing height for the wide rectangles $L_{wide}^{(\ell)}$ on stack ℓ and the narrow rectangles $L_{narrow}^{(\ell)}$ into a strip of width c_ℓ is bounded by $m_\ell(1 + 2\alpha)$. The rounding over the groups and the selection process above generates instances $(\bar{L}_{wide}^{(\ell)}, \bar{L}_{narrow}^{(\ell)})$ with rectangles of height 1 or, equivalently, with sets \bar{A}_ℓ of selected items.

LEMMA 2.4. *The set \bar{A}_ℓ can be packed into $m_\ell(1 + 6\alpha) + 5/\delta^2$ bins of capacity c_ℓ for each $\ell = 1, \dots, t$.*

Proof. First, we compare the corresponding strip packing set $\bar{L}_{wide}^{(\ell)}$ with $L_{sup}^{(\ell)}$ (where all original wide rectangles in part j on the stack ℓ are rounded up to the widest width in part j). To compare the solutions let us consider the following relation \leq introduced by Kenyon and Remila [10]. Given any list L of rectangles, first build a stack packing as described above. Let $STACK(L)$ be the polygonal region in the plane covered by the stack packing for L . Then two sets L and L' of rectangles (possibly with different numbers of rectangles) satisfy $L \leq L'$, if and only if $STACK(L)$ is contained in $STACK(L')$. This relation implies also the following inequalities [10]:

$$\begin{aligned} LIN_{frac-strip}(L) &\leq LIN_{frac-strip}(L') \\ AREA(L) &\leq AREA(L') \end{aligned}$$

where $LIN_{frac-strip}(L)$ is the minimum fractional strip packing height for instance L and $AREA(L)$ the total area of all rectangles in L . In our algorithm we have $\bar{L}_{wide}^{(\ell)} \leq L_{sup}^{(\ell)}$. This implies $LIN_{frac-strip}(\bar{L}_{wide}^{(\ell)}) \leq LIN_{frac-strip}(L_{sup}^{(\ell)})$. For the narrow rectangles we have $AREA(\bar{L}_{narrow}^{(\ell)}) \leq Area(\ell) = AREA(L_{narrow}^{(\ell)})$.

The total height of the strip packing generated by the algorithm of Kenyon and Remila [10] for the instance $\bar{L}_{wide}^{(\ell)} \cup \bar{L}_{narrow}^{(\ell)}$ is bounded by $\max[AREA(L_{wide}^{(\ell)} \cup L_{narrow}^{(\ell)}), LIN_{frac-strip}(L_{wide}^{(\ell)})](1 + \delta)/(1 - \delta) + 4/\delta^2 + 1$.

Since $AREA(L) \leq LIN_{frac-strip}(L)$ for each instance L , we obtain as height of the strip packing $LIN_{frac-strip}(L_{wide}^{(\ell)} \cup L_{narrow}^{(\ell)})(1 + 3\delta) + 5/\delta^2 \leq m_\ell(1 + 2\alpha)(1 + 3\delta) + 5/\delta^2 \leq m_\ell(1 + 6\alpha) + 5/\delta^2$ for $\delta = \alpha < 1/6$. In the first inequality we use that the fractional strip packing height for $L_{wide}^{(\ell)} \cup L_{narrow}^{(\ell)}$ is bounded by $\sum_{j=1}^{H_\ell} \tilde{y}_j^{(\ell)} \leq m_\ell(1 + 2\alpha)$. Since each rectangle in $\bar{L}_{wide}^{(\ell)} \cup \bar{L}_{narrow}^{(\ell)}$ has height 1, the strip packing can be converted easily into a bin packing. \square

In our algorithm we use the strip packing algorithm [10] for each instance $\bar{L}_{wide}^{(\ell)} \cup \bar{L}_{narrow}^{(\ell)}$ and strip width c_ℓ . The running time can be bounded by $poly(n, 1/\epsilon)$ [10].

2.5 Shifting Technique. Note that $|K_\ell \cup L_\ell \cup M_\ell| \leq 3/\delta^2 + 2 \leq 4/\delta^2$ and that we have $m_\ell \geq 1/\delta^3$ many bins. Using Lemma 2.4, $X_\ell = \bar{A}_\ell \cup (K_\ell \cup L_\ell \cup M_\ell)$ can be packed into $m_\ell(1 + 6\alpha) + 9/\delta^2$ bins.

LEMMA 2.5. *We can select a subset $X'_\ell \subset X_\ell$ with profit at least $(1 - 9\delta)(1 - 7\alpha)profit(X_\ell)$ that can be packed into m_ℓ bins.*

Proof. Using that $1/\delta$ is an integer and few empty bins (if necessary), the number of bins is equal to $m_\ell + \lceil 6\alpha m_\ell \rceil + 9/\delta^2$. Since $6\alpha m_\ell + 1 \leq 7\alpha m_\ell$ (using $m_\ell \geq 1/\delta^3 \geq 1/\alpha$), the profit of $\lceil 6\alpha m_\ell \rceil$ bins is at most 7α times the profit $profit(X_\ell)$ of the selected items. To see this take a packing of X_ℓ into the bins and split the bins into groups with $\lceil 6\alpha m_\ell \rceil$ bins (the last group has eventually less bins). We have at least $\frac{m_\ell + \lceil 6\alpha m_\ell \rceil}{\lceil 6\alpha m_\ell \rceil} \geq \lceil 1/(7\alpha) \rceil$ many large groups (using that $1/\alpha$ is integral and some math. calculations). One of these groups has a profit of at most $\frac{1}{\lceil 1/(7\alpha) \rceil} profit(X_\ell) \leq 7\alpha profit(X_\ell)$. Removing this group of bins and the corresponding items gives $m_\ell + 9/\delta^2$ bins with profit at least $(1 - 7\alpha)profit(X_\ell)$. In a similar way we can prove that the profit of $9/\delta^2$ bins is at most $9\delta(1 - 7\alpha)profit(X_\ell)$. Removing also this group of bins and the corresponding items gives an item set X'_ℓ with profit at least $(1 - 9\delta)(1 - 7\alpha)profit(X_\ell)$ that can be packed into m_ℓ bins. \square

The entire approximation algorithm works as follows:

- (1) Solve the linear program approximately whose objective value is at least $(1 - 3\alpha)$ times the optimum LP value (see Section 2.1).

- (2) Build t stacks of wide rectangles and sets with narrow rectangles, split the stacks into $1/\delta^2$ parts and round the rectangles over the groups. Then select the items \bar{A}_ℓ via solving fractional knapsack problems and store the fractional items $K_\ell \cup L_\ell \cup M_\ell$ (see Section 2.2 and 2.3).
- (3) Use the strip packing algorithm by Kenyon and Remila for each group to pack the items \bar{A}_ℓ into $m_\ell(1 + 6\alpha) + 5/\delta^2$ bins (see Section 2.4).
- (4) Apply the shifting strategy to select subsets $X'_\ell \subset X_\ell = \bar{A}_\ell \cup (K_\ell \cup L_\ell \cup M_\ell)$ that can be packed into m_ℓ bins (see Section 2.5).

Using Lemma 2.3 we have that the profit $\text{profit}(\bigcup_\ell X_\ell) \geq (1 - 3\alpha)OPT(LP)$. That implies using Lemma 2.5 that the selected subset $\bigcup_\ell X'_\ell$ over all groups has total profit at least $(1 - 9\delta - 10\alpha)OPT(LP)$ and can be packed into the bins. Using $\delta = \alpha \leq (1/19)\epsilon$ we obtain a feasible solution with profit at least $(1 - \epsilon)OPT(LP) \geq (1 - \epsilon)OPT(A, B)$. To satisfy that $1/\delta$ is integral, simply choose $\delta = 1/\lceil 19/\epsilon \rceil$.

3 Instances with Few Bins.

Let us consider here the case of a constant number γ of bins. Suppose that the bins are sorted in increasing order of their capacities: $c(b_1) \leq c(b_2) \leq \dots \leq c(b_\gamma)$. Let $APP(A, B)$ be an approximate value obtained by a fast algorithm (e.g. using *Greedy*($\epsilon'/2$) we can get $APP(A, B) \geq (1/2 - \epsilon'/4)OPT(A, B)$). One nice technique is to round the profit of items a_i with $\text{profit}(a_i) > (\rho/\gamma)OPT(A, B)$ down to the next multiple of $(\epsilon'/(\gamma\rho))2(1+\epsilon')APP(A, B)$ (here ρ is a constant that will be specified later). Rounding all such items generates a profit loss of at most $2\epsilon'(1+\epsilon')APP(A, B) \leq 2\epsilon'(1+\epsilon')OPT(A, B)$, since there are at most γ/ρ items with high profit in any feasible solution. Notice that $2(1+\epsilon')APP(A, B) \geq 2(1+\epsilon')(1/2 - \epsilon'/4)OPT(A, B) \geq OPT(A, B)$ for any $\epsilon' \leq 1$.

Furthermore, there are at most γ/ρ items in the instance with size $\text{size}(a_i) \leq \rho c(b_1)$ and profit $\text{profit}(a_i) > (\rho/\gamma)OPT(A, B)$ (otherwise there would be a solution with profit larger than $OPT(A, B)$). Therefore, there are at most γ/ρ items of profit larger than $(\rho/\gamma)2(1+\epsilon')APP(A, B) \geq (\rho/\gamma)OPT(A, B)$ and size at most $\rho c(b_1)$. Let $SmHi$ be the set of these items. Rounding this set of items as above generates also a profit loss of at most $2\epsilon'(1+\epsilon')OPT(A, B)$. In addition for each rounded profit value $\text{round}(k) = k[(\epsilon'\rho)/\gamma]2(1+\epsilon')APP(A, B)$ with $k \in \{1/\epsilon', \dots, \gamma/(\epsilon'\rho)\}$ we store the smallest (at most) $(1/k)(\gamma/(\epsilon'\rho)) \leq \gamma/\rho$ items with size $\text{size}(a_i) > \rho c(b_1)$ and rounded profit value $\text{round}(k)$. We denote with $A(k)$ the set of items for

each k . Note that $A(k)$ contains by definition at most $(1/k)(\gamma/(\epsilon'\rho))$ many items and this is the maximum number of items with profit value $\text{round}(k)$ in any feasible solution. This implies that there is only a small number $O([\gamma/(\epsilon'\rho)] \log[\gamma/(\epsilon'\rho)])$ of important items with high profit in the instance. Using $\epsilon', \rho = O(\epsilon)$, the number of these items is at most $O([\gamma/\epsilon^2] \log[\gamma/\epsilon^2])$.

In the following we show how an optimum solution can be replaced with an approximate solution with additional structure (i.e. where the items with high profit are selected only from $\bigcup_k A(k) \cup SmHi$). Consider an optimum solution $Opt \subset A$. First, take the set $Opt_{s,h} \subset Opt$ of items with size at most $\rho c(b_1)$ and high profit. Since $Opt_{s,h} \subset SmHi$ (these are the only small items with high profit), we do not replace these items. Second, let $Opt(k)$ be the items with rounded profit value $\text{round}(k)$ and size larger than $\rho c(b_1)$.

For each rounded value $\text{round}(k)$, replace now the items in $Opt(k)$ with the items from $A(k)$ with the smallest size. These items are by definition still large relative to bin b_1 . Let us order the items $a_1, \dots, a_{\ell(k)}$ in $Opt(k)$ by their sizes: $\text{size}(a_1) \leq \dots \leq \text{size}(a_{\ell(k)})$. Let us order also the items in $A(k)$ in non-increasing order of sizes: $\text{size}(\tilde{a}_1) \leq \dots \leq \text{size}(\tilde{a}_{m(k)})$. Clearly $\ell(k) \leq m(k)$ and $\text{size}(\tilde{a}_i) \leq \text{size}(a_i)$ for $i = 1, \dots, \ell(k)$. In each step we replace a_i by \tilde{a}_i . Let us denote with $A'(k)$ the set of the first $\ell(k)$ items from $A(k)$. Using the property $\text{size}(\tilde{a}_i) \leq \text{size}(a_i)$, we obtain also a feasible solution. Furthermore, the remaining capacity of each bin (after replacing the high profit items) is larger than or equal to the remaining capacity in the optimum solution. Let \overline{Opt} be the union of the set $Opt_{s,h}$ and set $\bigcup_k A'(k)$. All other remaining items in $Opt_{rem} = Opt \setminus (Opt_{s,h} \cup \bigcup_k Opt(k))$ have a small profit at most $(\rho/\gamma)2(1+\epsilon')APP(A, B)$. Let $Opt^{high} = (Opt_{s,h} \cup \bigcup_k Opt(k))$ be the set of all items with high profit in the optimum solution.

LEMMA 3.1. *The set $[Opt \setminus (\bigcup_k Opt(k))] \cup \bigcup_k A'(k)$ is a feasible solution of MKP and the rounded profit value of this solution is at least $OPT(A, B) - 2(1 + \epsilon')\epsilon'APP(A, B)$.*

Proof. By the argument above, the items in $Opt(k)$ can be replaced by $A'(k)$. Using the property above about the sizes, this gives directly a feasible solution of MKP. Notice that there are at most γ/ρ items with high profit in any feasible solution (otherwise we have a solution with profit larger than $2(1 + \epsilon')APP(A, B) \geq OPT(A, B)$). This implies the procedure above replaces at most γ/ρ items. The profit loss from each item is at most $[(\epsilon'\rho)/\gamma]2(1+\epsilon')APP(A, B)$ (since we round these items down). Therefore, the total profit loss is bounded by $2(1 + \epsilon')\epsilon'APP(A, B)$. \square

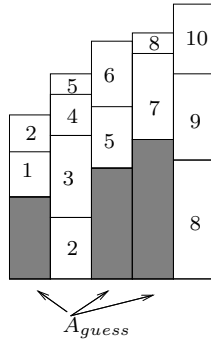


Figure 3: Choice of A_{guess} and fractional knapsack solution.

In our algorithm we guess a subset A_{guess} of the set $\bigcup_k A(k) \cup SmHi$ with at most γ/ρ items. Then we test whether this set A_{guess} fits into the bins. If the set does not fit into the bins we discard the subset. Note that one of the subsets A_{guess} corresponds to the set \overline{Opt} . We use a classical fractional knapsack algorithm to choose the remaining items. We simply take here all remaining items with profit at most $(\rho/\gamma)2(1+\epsilon')APP(A, B)$ and set the capacity of the knapsack to $cap - size(A_{guess})$ where $cap = \sum_{i=1}^{\gamma} c(b_i)$. Let A_s be the computed set of items with at most one fractional item. Suppose that the last item in A_s is fractional (if there is any fractional item). Afterwards we distribute the computed set A_s to the bins. By this process, we have to split at most $\gamma - 1$ many items (one item for each bin b_i with $i < \gamma$) and have at most one fractional item in the last bin b_γ . Let $Split_s \subset A_s$ be the set of split and fractional items. In Figure 5 we have a set A_{guess} with three items placed into three of the five bins. The set $A_s = \{1, \dots, 10\}$ obtained by the fractional knapsack algorithm fits into the remaining space. Furthermore, the distribution process had to split the items 2, 5 and 8.

Consider now the case where $A_{guess} = \overline{Opt}$. Since the total capacity $cap - size(\overline{Opt}) \geq cap - size(Opt_{high})$ and all items in Opt_{rem} have small profit, the profit of the selected set is $profit(A_s) \geq profit(Opt_{rem})$ and the total profit of the items in $Split_s$ is at most $2\rho(1+\epsilon')APP(A, B) \leq 2\rho(1+\epsilon')OPT(A, B)$. Our algorithm generates in this way a packing for $A_{guess} \cup (A_s \setminus Split_s)$. Finally, we take among all feasible choices a solution with highest profit. The entire algorithm for instances with γ bins and accuracy $\epsilon > 0$ can be described as follows:

- (0) Set $\epsilon' = \rho = \epsilon/5$.
- (1) Compute an approximate solution for the instance with value $APP(A, B) \geq (1/2 - \epsilon'/4)OPT(A, B)$.

- (2) Compute the set $SmHi$ and the sets $A(k)$ of items with high profit.
- (3) For each subset A_{guess} of $\bigcup_k A(k) \cup SmHi$ with at most γ/ρ items
 - (3.1) test whether A_{guess} fits into the bins; if not, then discard the subset A_{guess} ,
 - (3.2) if yes, then take a feasible assignment of A_{guess} to the bins,
 - (3.3) use a fractional knapsack algorithm to choose the remaining items with small profit and knapsack capacity $\sum_{i=1}^{\gamma} c(b_i) - size(A_{guess})$,
 - (3.4) distribute the computed set A_s to the bins and remove the items in $Split_s$.
- (4) Take a feasible subset A_{guess} with maximum total profit $profit(A_{guess} \cup (A_s \setminus Split_s))$.

LEMMA 3.2. *The algorithm above computes a packing into γ bins with profit at least $(1 - \epsilon)OPT(A, B)$, and has a running time $2^{O((\gamma/\epsilon)\log(\gamma/\epsilon))} \cdot n$.*

Proof. The total profit of $A_{guess} = \overline{Opt}$ and $A_s \setminus Split_s$ is at least $OPT(A, B) - 2(\epsilon' + \rho)(1 + \epsilon')OPT(A, B)$. Using $\epsilon' = \rho = \epsilon/5$, we get $2(\epsilon' + \rho)(1 + \epsilon') \leq \epsilon$ for any $\epsilon > 0$. Therefore, we obtain a solution with profit value at least $(1 - \epsilon)OPT(A, B)$. The running time of the algorithm is dominated by guessing the subset and testing whether a subset fits into the bins. Since $\epsilon' = \rho = O(\epsilon)$, we have to store only at most $O(\gamma/\epsilon^2 \log(\gamma/\epsilon^2))$ many items. The number of subsets of this set with cardinality at most γ/ρ is at most $[O(\gamma/\epsilon^2 \log(\gamma/\epsilon^2))]^{\gamma/\rho} \leq 2^{O((\gamma/\epsilon)\log(\gamma/\epsilon))}$. The number of assignments for one set A_{guess} to γ bins is at most $\gamma^{\gamma/\rho} \leq 2^{(\gamma/\rho)\log \gamma}$. Counting also the running time of $O(n)$ for the fractional knapsack algorithm we obtain a total running time for instances with γ bins of $2^{O((\gamma/\epsilon)\log(\gamma/\epsilon))} \cdot n$. \square

For $\gamma = O(1/\epsilon^4)$, the running time of the algorithm can be bounded by $2^{O(\log(1/\epsilon)/\epsilon^5)} \cdot n$.

4 General Instance.

In this section we consider now general instances of MKP. First, we have to consider two cases: when the number m of bins is smaller than or equal to the constant $2\delta^{-4}$, and when it is larger than $2\delta^{-4}$ (where $\delta = O(\epsilon)$ is specified later). In the first simpler case, we can use the approach described in Section 3 and obtain a parameterized polynomial time approximation scheme with running time $2^{O(\log(1/\epsilon)/\epsilon^5)}n$. Suppose from now that the number of bins is larger than $2\delta^{-4}$. In the first step of our scheme we modify the structure of the bins.

4.1 Modifying the Bins. Let us order the bins according to their capacities: $c(b_1) \leq c(b_2) \leq \dots \leq c(b_m)$. Suppose that $1/\delta$ is integral. Then, define $k = \lfloor \frac{m-\delta^{-4}}{\delta^{-3}} \rfloor$ as the number of bin groups of cardinality δ^{-3} in $B \setminus \{b_{m-\delta^{-4}+1}, \dots, b_m\}$. That means that we have k groups with δ^{-3} bins, one group with δ^{-4} bins and possibly one group with less than δ^{-3} bins. Let B_1 be the set of first $k\delta^{-3}$ bins (i.e. $B_1 = \{b_1, \dots, b_{k\delta^{-3}}\}$) and let $B_2 = B \setminus B_1$ be the remaining bins with higher capacities (i.e. $B_2 = \{b_{k\delta^{-3}+1}, \dots, b_m\}$).

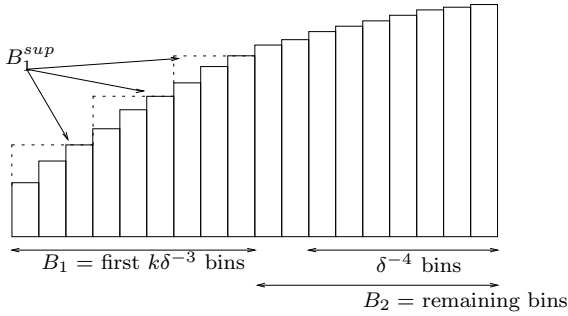


Figure 4: Bin structure before rounding.

Now round up the capacity of each bin in the first k groups of bins to the largest capacity of a bin in their group. In other words: B_1^{sup} is a set of k groups with δ^{-3} bins of capacity $\max_{j=1, \dots, \delta^{-3}} c(b_{i\delta^{-3}+j})$ for $i = 0, \dots, k-1$. In a similar way define B_1^{inf} as the set of k groups of δ^{-3} bins of capacity $\min_{j=1, \dots, \delta^{-3}} c(b_{i\delta^{-3}+j})$ for $i = 0, \dots, k-1$. This implies that each capacity in B_1^{sup} and B_1^{inf} occurs at least δ^{-3} times. An example with two groups of bins B_1 and B_2 is given in Figure 4. The rounding of the first k groups with δ^{-3} bins is indicated by the dashed lines. Note that B_2 contains between δ^{-4} and $2\delta^{-4}$ bins. In the following we show that the k .th group of bins can be eliminated. The main reason is that there are at least δ^{-4} additional bins with larger capacities. Let $Group_k = \{b_{(k-1)\delta^{-3}+1}, \dots, b_{k\delta^{-3}}\}$ be the bins of the k .th group.

LEMMA 4.1. *We can transform the optimum solution for an instance (A, B) such that the k .th group of bins is not used and the profit loss is at most $\delta \cdot OPT(A, B)$.*

Proof. Consider the δ^{-4} bins with largest capacities in the instance and take an optimum solution with value $OPT(A, B)$. Then we can divide these bins into $1/\delta$ many groups with δ^{-3} bins. There is at least one group where the optimum profit of all items placed in the corresponding bins is at most $\delta \cdot OPT(A, B)$ (otherwise we have a solution with profit larger than $OPT(A, B)$).

In other words we can eliminate the corresponding items and lose profit at most $\delta OPT(A, B)$. In addition we can move all items placed in the k .th group into the space left. After that we have a packing for $(A, B \setminus Group_k)$ with profit at least $(1 - \delta)OPT(A, B)$. \square

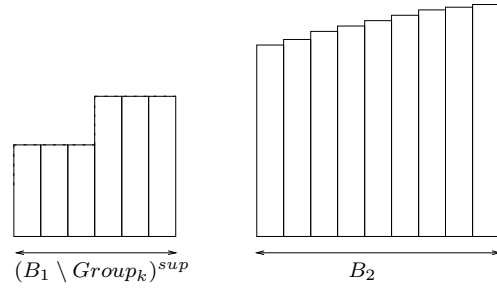


Figure 5: Modified bin structure.

Now we compare B_1^{sup} and B_1^{inf} . We take in our algorithm as instance the bin set $B' = B'_1 \cup B_2$ where $B'_1 = (B_1 \setminus Group_k)^{sup}$. The modified bin structure $B' = B'_1 \cup B_2$ for the instance in Figure 4 is illustrated in Figure 5. Lemma 4.2 implies that we can compute an approximate solution for $(A, B'_1 \cup B_2)$ and place the items into the bins $(B_1 \setminus Group_1) \cup B_2$.

LEMMA 4.2. *The profit $OPT(A, B') \geq (1 - \delta)OPT(A, B)$. In addition, $B'_1 \leq (B_1 \setminus Group_1)^{inf} \leq (B_1 \setminus Group_1)$*

Proof. The optimum profit $OPT(A, B'_1 \cup B_2)$ is larger than $OPT(A, (B_1 \setminus Group_k) \cup B_2)$, since the bin capacities in $B'_1 = (B_1 \setminus Group_k)^{sup}$ are larger or equal to the bin capacities in $(B_1 \setminus Group_k)$. In addition, using the Lemma 4.1, we have $OPT(A, (B_1 \setminus Group_k) \cup B_2) \geq (1 - \delta)OPT(A, B)$. This implies $OPT(A, B'_1 \cup B_2) \geq (1 - \delta)OPT(A, B)$. For the second statement, $\max_{j=1, \dots, \delta^{-3}} c(b_{i\delta^{-3}+j}) \leq \min_{j=1, \dots, \delta^{-3}} c(b_{(i+1)\delta^{-3}+j})$ implies the first inequality. For the second inequality, notice that $B^{inf} \leq B$ for any set B with $k\delta^{-3}$ bins. \square

After this step we have now a modified instance (A, B') with $B' = B'_1 \cup B_2$ where B'_1 consists of $(k-1)$ groups of δ^{-3} bins of the same capacity and B_2 consists of one group with $\leq \delta^{-3} + \delta^{-4} - 1 \leq 2\delta^{-4}$ bins with larger capacities. In other words, B_2 has at most $\gamma \leq 2\delta^{-4}$ bins and B'_1 is a set of bins with many similar capacities as in Section 2. The first modification step can be done in $O(m \log m) = O(n \log n)$ time.

4.2 Restricting the Approximate Solution.

Suppose that B'_1 has now bins with t different capacities c_1, \dots, c_t and $m_\ell \geq 1/\delta^3$ many bins of capacity c_ℓ . Let $APP(A, B)$ be the profit of the greedy

algorithm *Greedy*($\epsilon'/2$) for instance (A, B) and constant accuracy ϵ' specified later (i.e. $APP(A, B) \geq (1/2 - \epsilon'/4)OPT(A, B)$). Consider now only items with large profit $> (\rho/\gamma)2(1 + \epsilon')APP(A, B)$ (where ρ is a constant specified later and γ is the number of bins in B_2). Notice that there are at most γ/ρ such items in any optimum solution (otherwise the profit would be larger than $OPT(A, B)$). Now round the profit of items with large profit down to the next multiple of $[(\epsilon'\rho)/\gamma]2(1 + \epsilon')APP(A, B)$ and consider their sizes:

Case A: If $size(a) \leq \rho c_{min}(B_2)$ then there are at most γ/ρ many items in the instance. Notice that we could pack all of them in B_2 . We denote with *SmHi* the set of small items.

Case B: Consider now the case with $size(a) > \rho c_{min}(B_2)$. We store for each rounded profit value $round(k) = k[(\epsilon'\rho)/\gamma]2(1 + \epsilon')APP(A, B)$ (with $k \geq 1/\epsilon'$) at most $(1/k)(\gamma/(\rho\epsilon'))$ smallest items with rounded profit value $round(k)$ and size $size(a) > \rho c_{min}(B_2)$. Let $A(k)$ be the stored set.

In total we store at most $|\bigcup_{k \geq 1/\epsilon'} A(k) \cup SmHi| \leq O(\gamma/(\epsilon'\rho) \log[\gamma/(\epsilon'\rho)])$ many items. Using $\epsilon', \rho = O(\epsilon)$, this number of items is at most $O(\gamma/\epsilon^2 \log[\gamma/\epsilon^2])$.

Consider an optimum solution Opt for instance (A, B') and denote with $Opt_{s,h}^{(1)}$ and $Opt_{s,h}^{(2)}$ the items with size at most $\rho c_{min}(B_2)$ and high profit that are placed in the bins in B'_1 and B_2 , respectively. Furthermore, let $Opt^{(1)}(k)$ and $Opt^{(2)}(k)$ be the items with size larger than $\rho c_{min}(B_2)$ and rounded profit value $round(k)$ that lie in B'_1 and B_2 . We replace the items from $Opt^{(1)}(k)$, $Opt^{(2)}(k)$ by subsets $A'_1(k), A'_2(k) \subset A(k)$ (replace the smallest item from $Opt^{(1)}(k) \cup Opt^{(2)}(k)$ by the smallest from $A(k)$, etc.). Let $Opt(k) = Opt^{(1)}(k) \cup Opt^{(2)}(k)$ and $A'(k) = A'_1(k) \cup A'_2(k)$. Note that $|Opt(k)| \leq 1/k[\gamma/\rho\epsilon']$; otherwise the profit of the optimum solution would be larger than $OPT(A, B)$. This implies that all high profit items are replaced by the stored items. Let $\overline{Opt}^{(2)}$ be the union of the set $Opt_{s,h}^{(2)}$ and set $\bigcup_k A'_2(k)$.

LEMMA 4.3. *The set $Mod = [Opt \setminus (\bigcup_k Opt(k))] \cup \bigcup_k A'(k)$ is also a feasible solution for (A, B') and the rounded profit value $profit_{round}(Mod)$ is at least $OPT(A, B') - 2(1 + \epsilon')\epsilon'OPT(A, B)$.*

For our algorithm we have first to guess a subset $A_{guess} \subset SmHi \cup \bigcup_k A(k)$ with high profit items for the bins in B_2 . This is done through enumeration. Notice that we have only a constant number (at most γ/ρ) of candidates to be placed into a constant number γ of bins. A feasible choice for a set A_{guess} and placement into B_2 for our example is given in Figure 6. The number of choices for the set A_{guess} is at most

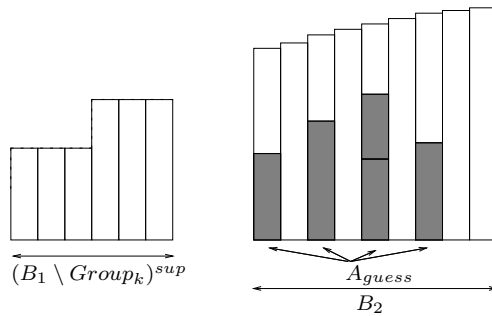


Figure 6: Choice of A_{guess} in B_2 .

$2^{O((\gamma/\rho) \log(\gamma/\epsilon^2))} = 2^{O(\log(1/\epsilon)/\epsilon^5)}$. The number of placements is again at most $2^{O(\gamma/\rho \log(\gamma))} = 2^{O(\log(1/\epsilon)/\epsilon^5)}$. In total we have to consider at most $2^{O(\log(1/\epsilon)/\epsilon^5)}$ choices and placements. Furthermore, there is a set A_{guess} that is equal to $\overline{Opt}^{(2)}$ corresponding to an optimum solution Opt . The main algorithm works now as follows:

- (1) Compute an approximate solution with value $APP(A, B) \geq (1/2 - \epsilon'/4)OPT(A, B)$ where $\epsilon' = \epsilon/12$.
- (2) Modify the structure of the bins: round the first k groups of bin capacities, eliminate group k and obtain the new instance $(A, B') = (A, (B_1 \setminus Group_k)^{sup} \cup B_2)$.
- (3) Compute the set *SmHi* and sets $A(k)$ of items with high profit that could be placed into B_2 .
- (4) For each subset (choice) $A_{guess} \subset SmHi \cup \bigcup_k A(k)$ of γ/ρ items
 - (4.1) test whether A_{guess} fits into the bins B_2 ; if not, then we discard the solution,
 - (4.2) if yes, we take a feasible placement of A_{guess} into B_2 and set up a linear program to select the remaining items (as described in Section 4.3),
 - (4.3) place the selected items into the bins $(B_1 \setminus Group_1) \cup B_2$.
- (5) Take a solution among all feasible choices A_{guess} with maximum total profit.

4.3 Modified Linear Program Relaxation. In this subsection we select the other items via a modified linear program. Let cap be the total capacity $\sum_{b_i \in B_2} c(b_i)$ of all bins in B_2 and let $size(A_{guess})$ be the total size of all selected items with high profit in B_2 . In the linear program below we use additional variables

z_i and as capacity for the fractional knapsack inequality the value $\overline{cap} = cap - size(A_{guess})$ and allow only small profit items (or pieces of them) with size $size(a_i) \leq \overline{cap}$ to be placed in B_2 . Suppose that the first $n' \leq n$ items have a small profit and fit into the knapsack as described above (otherwise reorder the items). Next we suppose that the next $n'' - n'$ items have either a small profit and size larger than \overline{cap} or have high profit and are not selected in A_{guess} (otherwise reorder the items). For these items we also assume that their sizes are at most $c_{max}(B'_1)$ (otherwise they do not fit into B'_1). All other items can be discarded in the enumeration step for A_{guess} . Notice that we implicitly allow to select items out of the second group of items for B'_1 but not for B_2 . Then the modified linear program has the following form:

$$\begin{aligned} \max \sum_i profit(a_i)x_i \\ \sum_{\ell, j: a_i \in C_j^{(\ell)}} y_j^{(\ell)} + z_i &= x_i && \text{for } i = 1, \dots, n' \\ \sum_{\ell, j: a_i \in C_j^{(\ell)}} y_j^{(\ell)} &= x_i && \text{for } i = n' + 1, \dots, n'' \\ \sum_{j=1}^{H_\ell} y_j^{(\ell)} &\leq m_\ell && \text{for } \ell = 1, \dots, t \\ \sum_{i=1}^{n'} size(a_i)z_i &\leq \overline{cap} \\ y_j^{(\ell)} &\geq 0 && \text{for } j = 1, \dots, H_\ell \\ &&& \text{and } \ell = 1, \dots, t \\ z_i &\in [0, 1] && \text{for } i = 1, \dots, n' \\ x_i &\in [0, 1] && \text{for } i = 1, \dots, n'' \end{aligned}$$

In the full paper we prove that the linear program above is a relaxation. In addition we show how to solve the modified linear program approximately and to generalize the rounding for the linear program and the selection process via fractional knapsack instances. In addition we show that the profit of the selected set $\bigcup_{\ell=1}^{t+1} X'_\ell$ together with the choice $A_{guess} = \overline{Opt}^{(2)}$ (that we guess in our algorithm) is at least $(1 - \epsilon)OPT(A, B)$. The running time of the algorithm to solve the linear program approximately is bounded by a polynomial $poly(n, 1/\epsilon)$ in the number n of items and $1/\epsilon$. The number of choices for the subset A_{guess} and the placements of A_{guess} into B_2 is at most $2^{O(\log(1/\epsilon)/\epsilon^5)}$. In total we obtain a running time of at most $2^{O(\log(1/\epsilon)/\epsilon^5)} \cdot poly(n) + O(m)$.

Acknowledgments. The author thanks Roberto Solis-Oba for many helpful discussions.

References

[1] E. Balas and E. Zemel: An algorithm for large zero-one knapsack problems, *Operations Research*, 28 (1980), 1130-1154.

[2] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, and R.E. Tarjan: Time bounds for selection, *Journal of Computer and System Sciences*, 7 (1973), 448-461.

[3] C. Chekuri and S. Khanna, A PTAS for the multiple knapsack problem, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, SODA 2000, 213-222.

[4] C. Chekuri and S. Khanna: A polynomial time approximation scheme for the multiple knapsack problem, *SIAM Journal on Computing*, 35 (2006), 713-728.

[5] F. Diedrich, K. Jansen, F. Pascual, and D. Trystram: Approximation algorithms for scheduling wit reservations, *Proceedings of Conference on High Performance Computing*, HIPC 2007, 297-307.

[6] D. Dor and U. Zwick: Selecting the median, *SIAM Journal on Computing*, 28 (1999), 1722-1758.

[7] R. Downey: Parametrized complexity for the skeptic, *Proceedings of IEEE Conference on Computational Complexity*, CCC 2003, 147-169.

[8] M.R. Fellows: Blow-ups, win/win's, and crown rules: some new directions in FPT, *Proceedings of Workshop on Graph Theoretical Concepts in Computer Science*, WG 2003, LNCS 2880, 1-12.

[9] M.D. Grigoriadis, L.G. Khachiyan, L. Porkolab and J. Villavicencio: Approximate max-min resource sharing for structured concave optimization, *SIAM Journal on Optimization*, 41 (2001), 1081-1091.

[10] C. Kenyon and E. Remila: Approximate strip packing, *Mathematics of Operations Research*, 25 (2000), 645-656.

[11] H. Kellerer: A polynomial time approximation scheme for the multiple knapsack problem, *Proceedings of Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, APPROX 1999, LNCS 1671, 51-62.

[12] H. Kellerer, U. Pferschy, and D. Pisinger: *Knapsack Problems*, Springer Verlag, Berlin, 2004.

[13] E.L. Lawler: Fast approximation algorithms for knapsack problems, *Mathematics of Operations Research*, 4 (1979), 339-356.

[14] J.K. Lenstra, D.B. Shmoys, and E. Tardos: Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, 24 (1990), 259-272.

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

[16] D. Marx: Parametrized complexity and approximation algorithms, *The Computer Journal*, 51 (2008), 60-78.

[17] S.A. Plotkin, D.B. Shmoys, and E. Tardos: Fast approximation algorithms for fractional packing and covering problems, *Mathematics of Operations Research*, 20 (1995), 257-301.

[18] M. Scharbrodt, A. Steger, and H. Weisser: Approximability of scheduling with fixed jobs, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, SODA 1999, 961-962.

[19] D.B. Shmoys and E. Tardos: An approximation algorithm for the generalized assignment problem, *Mathematical programming*, Series A, 62 (1993), 461-474.