A PTAS for the Multiple Knapsack Problem

Chandra Chekuri* Sanjeev Khanna†

Abstract

The *Multiple Knapsack* problem (MKP) is a natural and well known generalization of the single knapsack problem and is defined as follows. We are given a set of n items and m bins (knapsacks) such that each item i has a profit $p(i)$ and a size $s(i)$, and each bin j has a capacity $c(j)$. The goal is to find a subset of items of maximum profit such that they have a feasible packing in the bins. MKP is a special case of the *Generalized Assignment* problem (GAP) where the profit and the size of an item can vary based on the specific bin that it is assigned to. GAP is APX-hard and a 2-approximation for it is implicit in the work of Shmoys and Tardos [26], and thus far, this was also the best known approximation for MKP. The main result of this paper is a polynomial time approximation scheme for MKP.

Apart from its inherent theoretical interest as a common generalization of the well-studied knapsack and bin packing problems, it appears to be the strongest special case of GAP that is not APX-hard. We substantiate this by showing that slight generalizations of MKP that are very restricted versions of GAP are APX-hard. Thus our results help demarcate the boundary at which instances of GAP become APX-hard. An interesting and novel aspect of our approach is an approximation preserving reduction from an arbitrary instance of MKP to an instance with $O(\log n)$ distinct sizes and profits.

1 Introduction

We study the following natural generalization of the classical knapsack problem:

Multiple Knapsack Problem (MKP)
INSTANCE: A pair $(\mathcal{B}, \mathcal{S})$ where $\mathcal{B}$ is a set of m bins (knapsacks) and $\mathcal{S}$ is a set of n items. Each bin $j \in \mathcal{B}$ has a capacity $c(j)$, and each item i has a size $s(i)$ and a profit $p(i)$.
OBJECTIVE: Find a subset $U \subseteq \mathcal{S}$ of maximum profit such that U has a feasible packing in $\mathcal{B}$.

The decision version of MKP is a generalization of the decision versions of both the knapsack and bin packing problems and is strongly NP-Complete. Moreover, it is an important special case of the *generalized assignment* problem where both the size and the profit of an item are a function of the bin:

Generalized Assignment Problem (GAP)[1]
INSTANCE: A pair $(\mathcal{B}, \mathcal{S})$ where $\mathcal{B}$ is a set of m bins (knapsacks) and $\mathcal{S}$ is a set of n items. Each bin $j \in \mathcal{B}$ has a capacity $c(j)$, and for each item i and bin j, we are given a size $s(i, j)$ and a profit $p(i, j)$.
OBJECTIVE: Find a subset $U \subseteq \mathcal{S}$ of maximum profit such that U has a feasible packing in $\mathcal{B}$.

GAP and its restrictions capture several fundamental optimization problems and have many practical applications in computer science, operations research, and related disciplines. There has been extensive research in both theoretical and empirical aspects. The Multiple Knapsack problem with its uniform sizes and profits is an important special case and also has several applications. The book on knapsack variants by Martello and Toth [24] has a chapter devoted to MKP. Also referred to as the loading problem in operations research, it models the problem of loading items into containers of different capacities such that container capacities are not violated. In many practical settings items could be more complex geometric objects, however the one dimensional case (MKP) is useful in its own right and has been investigated extensively [12, 9, 10, 21, 22, 23, 4].

In addition to their practical interest, knapsack, bin packing, and related problems have attracted much theoretical attention for their simplicity and elegance, and their study has led to much interest in approximation algorithms. Though knapsack and bin packing have an FPTAS (asymptotic for bin packing), GAP, a strong generalization of both, however is APX-hard and only a 2-approximation exists. In fact some very special cases of GAP can be shown to be APX-hard. In particular we can show that for arbitrarily small $\delta > 0$ (can even be a function of n) the problem remains APX-hard on the following very restricted set of instances: for each item i and machine j, $p(i, j) = 1$, and $s(i, j) = 1$ or $s(i, j) = 1+\delta$, and bin capacities are all identical. The complementary case where item sizes do not vary across bins but profits do, can also be shown to be APX-hard for a similar restricted setting. In light of this, it is particularly interesting to understand the complexity of MKP where profits and sizes of an item are independent of the bin but the item sizes and profits as well as bin capacities may take any arbitrary values. Establishing a PTAS would show a very fine separation between cases that

*Bell Labs, 600 Mountain Ave, Murray Hill, NJ 07974. E-mail: chekuri@research.bell-labs.com.

†Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104. This work was done when the author was at Bell Labs, Murray Hill. Email : sanjeev@cis.upenn.edu.

[1] GAP has also been defined in the literature as a (closely related) minimization problem (see [26]). In this paper, following [24], we refer to the maximization version of the problem as GAP and refer to the minimization version as Min GAP.

are APX-hard and those that have a PTAS. Until now, the best known approximation ratio for MKP was a factor of 2 derived from the approximation for GAP.

Our main result: In this paper we resolve the approximability of MKP by obtaining a PTAS for it. It can be easily shown via a reduction from the Partition problem that MKP does not admit a FPTAS even if $m = 2$. A special case of MKP is when all bin capacities are equal. It is relatively straightforward to obtain a PTAS for this case using ideas from approximation schemes for knapsack and bin packing [11, 3, 15]. However the general case with *different* bin capacities is a non-trivial and challenging problem. Our paper contains two new technical ideas. Our first idea concerns the set of items to be packed in a knapsack instance. We show how to guess, in polynomial time, *almost all* the items that are to be packed in a given knapsack instance. In other words we can identify an item set that has a feasible packing and profit at least $(1-\epsilon)\mathrm{OPT}$. This is in contrast to earlier schemes for variants of knapsack [11, 2, 5] where only the $1/\epsilon$ most profitable items are guessed. An easy corollary of our strategy is a PTAS for the identical bin capacity case, the details of which we point out later. The strengthened guessing plays a crucial role in the general case in the following way. As a byproduct of our guessing scheme we show how to round the sizes of items to at most $O(\log n/\epsilon^2)$ *distinct* sizes. An immediate consequence of this is a quasi-polynomial time algorithm to pack items into bins using standard dynamic programming. Our second set of ideas shows that we can exploit the restricted sizes to pack, in *polynomial* time, a subset of the item set that has at least a $(1-\epsilon)$ fraction of the profit. Approximation schemes for number problems are usually based on rounding instances to have a fixed number of distinct values. In contrast, MKP appears to require a logarithmic number of values. We believe that our ideas to handle logarithmic number of distinct values will find other applications. Figure 1 summarizes the approximability of various restrictions of GAP.

Related work: MKP is closely related to knapsack, bin packing, and GAP. A very efficient FPTAS exists for the knapsack problem; Lawler [17], based on ideas from [11], achieves a running time of $O(n \log 1/\epsilon + 1/\epsilon^4)$ for a $(1+\epsilon)$ approximation. An asymptotic FPTAS is known for bin packing [3, 15]. Recently Kellerer [16] has independently developed a PTAS for the special case of the MKP where all bins have *identical* capacity. As mentioned earlier, this case is much simpler than the general case and falls out as a consequence of our first idea. The generalized assignment problem, as we phrased it, seeks to maximize profit of items packed. This is natural when viewed as a knapsack problem (see [24]). The minimization version of the problem, referred to as Min GAP and also as the cost assignment problem, seeks to assign all the items while *minimizing* the sum of the costs of assigning items to bins. In this version, item i incurs a cost $w(i,j)$ if assigned to bin j instead of a obtaining a profit. Since the feasibility of assigning all items is itself an NP-Complete problem we need to relax the bin capacity constraints. An (α, β) bi-criteria approximation algorithm for Min GAP is one that gives a solution with cost at most αC and with bin capacities violated by a factor of at most β where C is the cost of an optimal solution that does not violate any capacity constraints. Work of Lin and Vitter [20] yields a $(1+\epsilon, 2+1/\epsilon)$ approximation for Min GAP. Shmoys and Tardos [26], building on the work of Lenstra, Shmoys, and Tardos [19], give an improved $(1, 2)$ approximation. Implicit in this approximation is also a 2-approximation for the profit maximization version which we sketch later. Lenstra et al. [19] also show that it is NP-hard to obtain a ratio $(1, \beta)$ for any $\beta < 3/2$. The hardness relies on a NP-Completeness reduction from 3-Dimensional Matching. Our APX-hardness for the maximization version, mentioned earlier, is based on a similar reduction but instead relies on APX-hardness of the optimization version of 3-Dimensional matching [14].

MKP is also related to two variants of variable size bin packing. In the first variant we are given a set of items and set of bin capacities $\mathcal{C}$. The objective is to find a feasible packing of items using bins with capacities restricted to be from $\mathcal{C}$ so as to minimize the sum of the capacities of the bins used. A PTAS for this problem was provided by Murgolo [25]. The second variant is based on a connection to multi-processor scheduling on uniformly related machines [18]. The objective is to assign a set of independent jobs with given processing times to machines with different speeds to minimize the makespan of the schedule. Hochbaum and Shymoys [8] gave a PTAS for this problem using a *dual* based approach where they convert the scheduling problem into the following bin packing problem. Given items of different sizes and bins of different capacities, find a packing of all the items into the bins such that maximum violation of the capacity of any bin is minimized. Bi-criteria approximations, where both capacity and profit can be approximated simultaneously, have been studied for several problems (Min GAP being an example mentioned above) and it is usually the case that relaxing both makes the task of approximation somewhat easier. In particular, relaxing the capacity constraints allows rounding of item sizes into a small number of distinct size values. In MKP we are neither allowed to exceed the capacity constraints nor the number of bins. This makes the problem harder and our result interesting.

Organization: Section 2 describes our PTAS for MKP. In Section 3, we show that GAP is APX-hard on very restricted classes of instances. We also indicate here a 2-approximation for GAP. In Section 4, we discuss a natural greedy algorithm for MKP and show that it gives a $(2+\epsilon)$-approximation even when item sizes vary with bins.

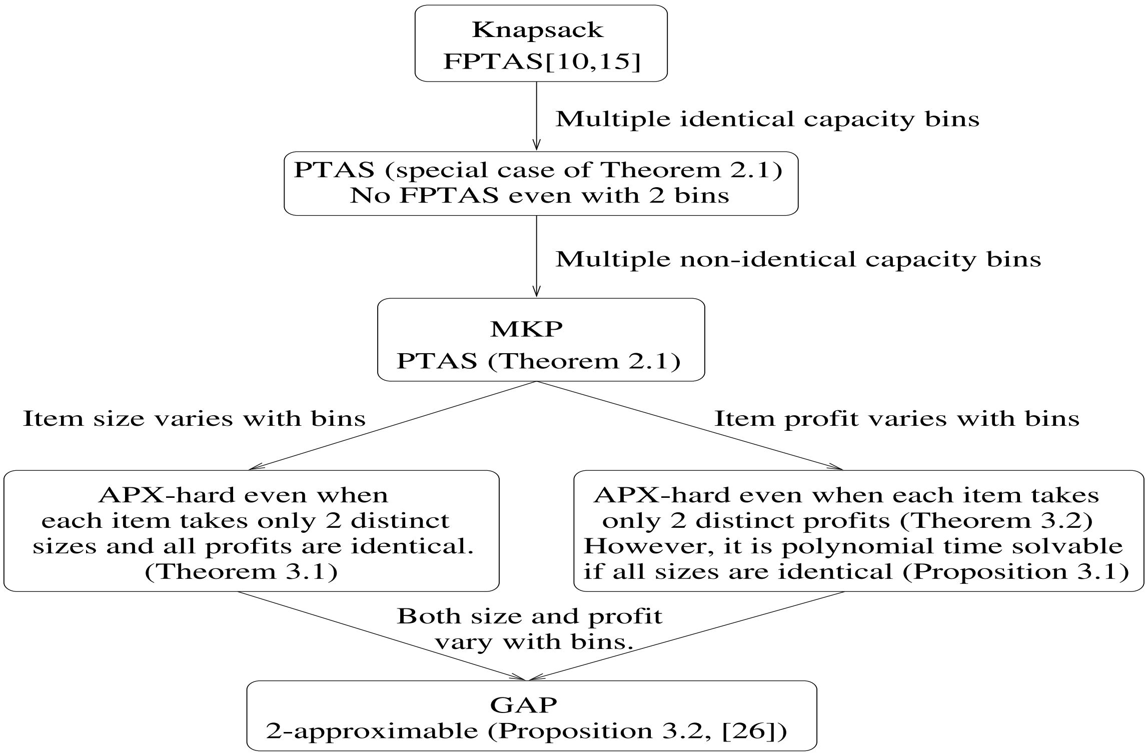

Figure 1: Complexity of Various Restrictions of GAP

2 A PTAS for the Multiple Knapsack Problem

We denote by OPT the value of an optimal solution to the given instance. Given a set Y of items, we use $p(Y)$ to denote $\sum_{y \in Y} p(y)$. The set of integers $0, 1, \ldots, k$ is denoted by $[k]$. Our problem is related to both the knapsack problem and the bin packing problem and some ideas used in approximation schemes for those problems will be useful to us. Our approximation scheme conceptually has the following two steps.

1. GUESSING ITEMS: Identify a set of items $U \subseteq \mathcal{S}$ such that $p(U) \geq (1-\epsilon)\text{OPT}$ and U has a feasible packing in $\mathcal{B}$.

2. PACKING ITEMS: Given a set U of items that has a feasible packing in $\mathcal{B}$, find a feasible packing for a set $U' \subseteq U$ such that $p(U') \geq (1-\epsilon)p(U)$.

The overall scheme is more involved since there is interaction between the two steps. The guessed items have some additional properties that are exploited in the packing step. We observe that both of the above steps require new ideas. For the single knapsack problem no previous algorithm identifies the full set of items to pack. Moreover testing the feasibility of packing a given set of items is trivial. However in MKP the packing step is itself quite complex and it seems necessary to decompose the problem as we did. Before we proceed with the details we show how the first step of guessing the items immediately gives a PTAS for the identical bin capacity case.

2.1 MKP with Identical Bin Capacities

Suppose we can guess an item set as in our first step above. We show that the packing step is very simple if bin capacities are identical. There are two cases to consider depending on whether m, the number of bins, is less than $\lceil 1/\epsilon \rceil$ or not. In the former case the number of bins can be treated as a constant and a PTAS for this case exists even for instances of GAP (implicit in earlier work [5]). Now suppose $m > \lceil 1/\epsilon \rceil$. We use the any of the known PTASs for bin packing and pack *all* the guessed items using at most $(1+\epsilon)m$ bins. We find a feasible solution by simply picking the m largest profit bins and discarding the rest along with their items. Here we use the fact that $m\epsilon \geq 1$ and that the bins are identical. It is easily seen that we get a $(1+\epsilon)$ approximation. We note that a different PTAS, without using our guessing step, can be obtained for this case by directly adapting the ideas used in approximation schemes for bin packing. The trick of using extra bins does not have a simple analogue when bin capacities are different and we need more ideas.

2.2 Guessing Items

Consider the case when items have the same profit (assume it is 1). Thus the objective is to pack as many items as possible. For this case it is easily seen that OPT is an integer in $[n]$. Further, given a guess for OPT, we can always pick the *smallest* (in size) OPT items to pack. Therefore there are only a polynomial number of guesses for the set of items to pack. This idea does not have a direct extension to non-uniform profits. However the useful insight is that when profits are identical we can pick the items in order of their sizes. In the rest of the paper we assume, for simplicity of notation, that $1/\epsilon$ and $\ln n$ are integers.

For the general case the first step in guessing items involves massaging the given instance into a more structured one that has few distinct profits. This is accomplished as follows.

1. Guess a value $\mathcal{O}$ such that $(1-\epsilon)\cdot \text{OPT} \leq \mathcal{O} \leq \text{OPT}$ and discard all items y where $p(y) < \epsilon\mathcal{O}/n$.
2. Scale all profits by $\epsilon\mathcal{O}/n$ such that they are in the range $[n/\epsilon]$.
3. Round *down* the profits of items to the nearest power of $(1+\epsilon)$.

The first two steps are similar to those in the FPTAS for the single knapsack problem. It is easily seen that at most an $O(\epsilon)$ fraction of the optimal profit is lost by our transformation. Summarizing:

LEMMA 2.1. *Given an instance $I = (\mathcal{B}, \mathcal{S})$ with n items and a value $\mathcal{O}$ such that $(1-\epsilon)\text{OPT}(I) \leq \mathcal{O} \leq \text{OPT}(I)$ we can obtain in polynomial time another instance $I' = (\mathcal{B}, \mathcal{S}')$ such that*

- $\mathcal{S}' \subseteq \mathcal{S}$.
- *For every $y \in \mathcal{S}'$, $p(y) = (1+\epsilon)^i$ for some $i \in [2\ln n/\epsilon]$.*
- $(1-\epsilon)\text{OPT}(I) \leq \frac{n}{\epsilon\mathcal{O}}\text{OPT}(I') \leq \text{OPT}(I)$.

The above lemma allows us to work with instances with $O(\ln n/\epsilon)$ distinct profits. We now show how we can use this information to guess the items to pack. Let $h \leq 2\ln n/\epsilon$ be the distinct number of profits in our new instance. We partition $\mathcal{S}$ into h sets $S_1, \ldots, S_h$ with items in each set having the same profit. Let U be the items chosen in some optimal solution and let $U_i = S_i \cap U$. Recall that we have an estimate $\mathcal{O}$ of the optimal value. If $p(U_i) \leq \epsilon\mathcal{O}/h$ for some i, we ignore the set S_i; no significant profit is lost. Hence we can assume that $\epsilon\mathcal{O}/h \leq p(U_i) \leq \mathcal{O}$ and approximately guess the value $p(U_i)$ for $1 \leq i \leq h$. More precisely, for each i we guess a value $k_i \in [h/\epsilon^2]$ such that $k_i(\epsilon^2\mathcal{O}/h) \leq p(U_i) \leq (k_i+1)(\epsilon^2\mathcal{O}/h)$.

A naive way of guessing the values $k_1, \ldots, k_h$ requires $\Theta(n^{O(\ln n/\epsilon^2)})$ time. We first show how the numbers k_i enable us to identify the items to pack and then show how the values $k_1, \ldots, k_h$ can in fact be guessed in polynomial time. Let a_i denote the profit of an item in S_i. Consider an index i such that $a_i \leq \epsilon\mathcal{O}/h$. Given the value k_i we order the items in S_i in increasing size values and pick the largest number of items from this ordered set whose cumulative profit does not exceed $k_i(\epsilon^2\mathcal{O}/h)$. If $a_i > \epsilon\mathcal{O}/h$ we pick the smallest number of items whose cumulative profit exceeds $k_i(\epsilon^2\mathcal{O}/h)$. The asymmetry is for technical reasons. The choice of items is thus completely determined by the choice of the k_i. For a tuple of values $k_1, \ldots, k_h$, let $U(k_1, \ldots, k_h)$ denote the set of items picked as described above.

LEMMA 2.2. *There exists a valid tuple $(k_1, \ldots, k_h)$ with each $k_i \in [h/\epsilon^2]$ such that $U(k_1, \ldots, k_h)$ has a feasible packing in $\mathcal{B}$ and $p(U(k_1, \ldots, k_h)) \geq (1-\epsilon)\mathcal{O}$.*

Proof. Let U be the items in some optimal solution and let $U_i = S_i \cap U$. Define k_i to be $\lfloor p(U_i)h/(\epsilon^2\mathcal{O})\rfloor$. The tuple so obtained will satisfy the required properties. □

As remarked earlier, a naive enumeration of all integer h-tuples will take quasi-polynomial time. The crucial observation is that the k_i's are not independent. They must satisfy the additional constraint that $\sum_i k_i \leq h/\epsilon^2$ since the total profit from all the sets $S_1, \ldots, S_h$ cannot exceed OPT. This constraint limits the number of tuples of relevance. We make use of the following claims.

CLAIM 2.1. *Let f be the number of g-tuples of non-negative integers such the sum of tuple coordinates is equal to d. Then $f = \binom{d+g-1}{g-1}$. If $d+g \leq \alpha g$ then $f = O(\alpha^{\alpha g})$.*

CLAIM 2.2. *Let $h \in [2\ln n/\epsilon]$. Then the number of h-tuples $(k_1, \ldots, k_h)$ such that $k_i \in [h/\epsilon^2]$ and $\sum_i k_i \leq \lceil h/\epsilon^2\rceil$ is $O(n^{O(1/\epsilon^3)})$.*

Using the restricted number of distinct profit values we can also reduce the number of *distinct sizes* in the given instance to $O(\ln n)$. This property will be crucial in packing the items. The basic idea is *shifting*, an idea that is used in approximation schemes for bin packing. Let A be a set of g items with identical profit but perhaps differing sizes. We order items in A in non-decreasing sizes and divide them into $t = (1+1/\epsilon)$ groups $A_1, \ldots, A_t$ with $A_1, \ldots, A_{t-1}$ containing $(g \text{ div } t)$ items each and A_t containing $(g \bmod t)$ items. We discard the items in A_{t-1} and for each $i < t-1$ we increase the size of every item in A_i to the size of the smallest item in A_{i+1}. Since A is ordered by size no item in A_i is larger than the smallest item in A_{i+1} for each $1 \leq i < t$. It is easy to see that if A has a feasible packing then the modified instance also has a feasible packing. We discard at most an ϵ fraction of the profit and the modified sizes have at most $2/\epsilon$ distinct values. Applying this to each profit class we obtain an instance with $O(\ln n/\epsilon^2)$ distinct size values.

LEMMA 2.3. *Given an instance $I = (\mathcal{B}, \mathcal{S})$ with n items we can obtain in polynomial time $v = n^{O(1/\epsilon^3)}$ instances $I_1, \ldots, I_v$ such that*

- *For $1 \leq j \leq v$, $I_j = (\mathcal{B}, \mathcal{S}_j)$ and $\mathcal{S}_j$ has items only from $O(\ln n/\epsilon)$ distinct profit values.*
- *For $1 \leq j \leq v$, $I_j = (\mathcal{B}, \mathcal{S}_j)$ and $\mathcal{S}_j$ has items only from $O(\ln n/\epsilon^2)$ distinct size values.*
- *There exists an index ℓ such that $\mathcal{S}_\ell$ has a feasible packing in $\mathcal{B}$ and $p(\mathcal{S}_\ell) \geq (1-\epsilon)\text{OPT}(I)$.*

We will assume for the next section that we have guessed the correct set of items and that they are partitioned into $O(\ln n/\epsilon^2)$ sets with each set containing items of the same size. We denote by U_i the items of the ith size value and by n_i the quantity $|U_i|$.

2.3 Packing Items

From Lemma 2.3 we obtain a restricted set of instances in terms of item profits and sizes. We also need some structure in the bins and we start by describing the necessary transformations.

2.3.1 Structuring the Bins

Assume without loss of generality that the smallest bin capacity is 1. We order the bins in increasing order of their capacity and partition them into *blocks* $B_0, B_1, \ldots, B_\ell$ such that block B_i consists of all bins x with $(1+\epsilon)^i \leq c(x) < (1+\epsilon)^{i+1}$. Let m_i denote the number of bins in block B_i.

DEFINITION 2.1. (Small vs. Large Blocks) *A block B_i of bins is called a* small *bin block if $m_i \leq 1/\epsilon$; it is called* large *otherwise.*

Let Q be the set of indices i such that B_i is small. Define Q' to be the set of $3\ln(1/\epsilon)/\epsilon$ largest indices in the set Q. Let B_Q and $B_{Q'}$ be the set of all bins in the blocks specified by the index sets Q and Q' respectively. The following lemma makes use of the property of geometrically increasing bin capacities.

LEMMA 2.4. *Let U be a set of items that can be packed in the bins B_Q. There exists a set $U' \subseteq U$ such that U' can be packed into the bins $B_{Q'}$, and $p(U') \geq (1-\epsilon) \cdot p(U)$.*

Proof. Fix some packing of U in the bins B_Q. Consider the largest $1/\epsilon$ bins in B_Q. One of these bins has a profit less than $\epsilon p(U)$. Without loss of generality assume its capacity is 1. We will remove the items packed in this bin and use it to pack items from smaller bins. Let B_i be the block containing this bin. Bins in block B_j where $j \in B_Q$ and $j < i - 2\ln(1/\epsilon)/\epsilon$ have a capacity less than ϵ^2. It is easy to verify that the total capacity of bins in small bin blocks with indices less than equal to j is at most 1. Since each of the first $1/\epsilon$ bins could be in different blocks the upper bound of $3\ln(1/\epsilon)/\epsilon$ blocks follows. □

Therefore we can retain the $3\ln(1/\epsilon)/\epsilon$ small bin blocks of large capacity and discard the rest. Therefore we assume that the given instance is modified to satisfy $|Q| \leq 3\ln(1/\epsilon)/\epsilon$. Then it follows that $|B_Q| \leq 3\ln(1/\epsilon)/\epsilon^2$. When the number of bins is fixed a PTAS is known (implicit in earlier work) even for the GAP. We will use ideas from that algorithm. For large bin blocks the advantage is that we can exceed the number of bins used by an ϵ fraction, as we shall see below. The main task is to integrate the allocation and packing of items between the different sets of bins.

For the rest of the section we assume that we have a set of items that have a feasible packing and we will implicitly refer to some fixed feasible packing as *the* optimal solution.

2.3.2 Packing Profitable Items into Small Bin Blocks

We guess here for each bin in B_Q, the $1/\epsilon$ most profitable items that are packed in it in the optimal solution. Therefore the number of guesses needed is $n^{O(\ln(1/\epsilon)/\epsilon^3)}$.

2.3.3 Packing Large Items into Large Bin Blocks

The second step is to select items and pack them in large bin blocks. We say that an item is packed as a *large item* if its size is at least ϵ times the capacity of the bin in which it is packed. Since the capacities of the blocks are increasing geometrically, an item can be packed as large in at most $f = \lceil \frac{\ln 1/\epsilon}{\epsilon} \rceil$ different blocks. Our goal is to guess for each item whether it is packed as large and if so in which large bin block. We do this approximately and the high level idea of the guessing is given below; full details are omitted from this version.

- Let n_i be the number of items of the ith size class U_i and let ℓ_i be the number packed as large in some optimal solution. If $\ell_i \leq \epsilon n_i$ we can ignore these large items from profit considerations. Otherwise we guess a number ℓ'_i such that $\ell_i(1-\epsilon) \leq \ell'_i \leq \ell_i$ which takes $O(1/\epsilon)$ guesses.
- Partition the ℓ'_i items into f_i groups where $f_i \leq f$ is the number of blocks in which the items of U_i can be packed as large. This takes $2^{O(\ln(1/\epsilon)/\epsilon^3)}$ guesses.
- Combining for all sizes results in $n^{O(\ln(1/\epsilon)/\epsilon^5)}$ guesses over all. Here is where we take advantage of the fact that our items come from only $O(\ln n/\epsilon^2)$ different size classes.

Suppose we have correctly assigned all large items to their respective bin blocks. We describe now a procedure for finding a feasible packing of these items. Here we ignore the potential interaction between items that are packed as large and those packed as small. We will show later that we can do so with only a slight loss in the approximation factor. We can focus on a specific block since the large items are now partitioned between the blocks. The abstract problem we have is the following. Given a collection of m bins with capacities in the range $[1, 1+\epsilon)$, and a set of n items with sizes in the range $(\epsilon, 1+\epsilon)$, decide if there is a feasible packing for them. It is easily seen that this problem is NP-hard. We obtain a relaxation by allowing use of *extra* bins to pack the items. However, we restrict the capacity of the extra bins to be 1. The following algorithm decides that either the given instance is infeasible or gives a packing with at most an additional ϵm bins of capacity 1.

- Let L be the set of items of size greater than 1. Order items of L in non-decreasing sizes and pack each item into the smallest bin available that can accommodate it.
- Disregard all the bins used up in the previous step since no other large item can fit into them.
- To pack the remaining items use a PTAS for bin packing, potentially using an extra ϵm bins of capacity 1.

We omit the full details of the algorithm but summarize the result in the following lemma.

LEMMA 2.5. *Given $m \geq 1/\epsilon$ bins of capacities in the range $[1, 1+\epsilon)$ and items of sizes in the range $(\epsilon, 1+\epsilon)$, there is an algorithm $\mathcal{A}$ that runs in $n^{O(1/\epsilon^2)}$ time, and provided the items have a feasible packing in the given bins, returns a feasible packing using at most ϵm extra bins of capacity 1.*

We eliminate the extra bins later by picking the m most profitable among them and discarding the items packed in the rest. The restriction on the size and number of extra bins is motivated by the elimination procedure. In order to use extra bins the quantity ϵm needs to be at least 1. This is the reason to distinguish between small and large bin blocks. For a large bin block B_i let E_i be the *extra* bins used in packing the large items. We note that $|E_i| \leq \epsilon |B_i| = \epsilon m_i$.

2.3.4 Packing the Remaining Items

The third and last step of the algorithm is to pack the remaining items which we denote by R. At this stage we have a packing of the $1/\epsilon$ most profitable items in each of the bins in B_Q (bins in small bin blocks) and a feasible packing of the large items in the rest of the bins (including the extra bins). For each bin $b_j \in \mathcal{B}$ let Y_j denote the set of items already packed into b_j in the first two steps. The item set R is packed via an LP approach. In particular we use the generalized assignment formulation with the following constraints.

1. Each remaining item must be assigned to some bin.
2. An item can y can be assigned to a bin b_j in a large bin block B_i only if $s(y) \leq \epsilon \cdot (1+\epsilon)^i$. In other words y should be small for all bins in B_i.
3. An item y can be assigned to a bin b_j in a small bin block only if $p(y) \leq \frac{\epsilon}{1+\epsilon} p(Y_j)$ and $|Y_j| \geq 1/\epsilon$.

Constraints 2 and 3 are based on the assumption that we have correctly guessed in the first two steps of the packing procedure. We make the formulation more precise now. Note that we only check for feasibility. The variable x_{ij} denotes the fraction of item i that is assigned to bin j. Let V be the set of item-bin pairs (i, j) such that i cannot be packed into b_j because of Constraints 2 and 3. The precise LP formulation is given below.

$$
\begin{array}{rcll}
\sum_j x_{ij} & = & 1 & \text{for each item} \\
\sum_i s(y_i) \cdot x_{ij} & \leq & c(b_j) & \text{for each bin} \\
x_{ij} & = & 0 & \text{if } (i,j) \in V \\
x_{ij} & = & 1 & \text{if } i \in Y_j \\
x_{ij} & \geq & 0 &
\end{array}
$$

LEMMA 2.6. *The LP formulation above has a feasible solution if the guesses for the item set, the items packed as large, and those packed in small bin blocks, are correct.*

Let x_{ij} be a feasible fractional solution to the above formulation. The analysis of Shmoys and Tardos [26] for the Min GAP shows that we can transform the fractional solution given by the x_{ij} to a 0-1 solution given by $\bar{x}_{ij}$ with the following properties.

1. If $x_{ij} = 0$ then $\bar{x}_{ij} = 0$ and if $x_{ij} = 1$ then $\bar{x}_{ij} = 1$.
2. For each bin b_j, either $\sum_i \bar{x}_{ij} \leq c(b_j)$ or there is an item $k(j)$ such that $\sum_{i \neq k(j)} \bar{x}_{ij} \leq c(b_j)$ and $x_{ik(j)} < 1$. We call this item $k(j)$ as the *violating* item for bin b_j.

Thus we can find an integral solution where each bin's capacity is exceeded by at most one item. Further the items assigned to the bins satisfy the constraints specified by V, that is $\bar{x}_{ij} = 0$ if $(i, j) \in V$. The integral solution to the LP also defines an allocation of items to each block. Let P_i be the total profit associated with all items assigned to bins in block B_i. Then clearly, $\mathcal{O} = \sum_{i \geq 0} P_i$. We however have an infeasible solution. Extra bins E_i are used for large bin blocks and bin capacities are violated in the rounded solution $\bar{x}_{ij}$. We modify this solution to create a feasible solution such that in each block we obtain a profit of at least $(1 - 3\epsilon)P_i$.

Large Bin Blocks: Let B_i be a large bin block and without loss of generality assume that bin capacities in B_i are in the range $[1, 1+\epsilon)$. By constraint 2 on the assignment, the size of any violating item in B_i is less than ϵ and there are at most m_i of them. For all $\epsilon < 1/2$ we conclude that at most $2\epsilon m_i$ extra bins of capacity 1 each are sufficient to pack all the violating items of B_i. Recall that we may have used ϵm_i extra bins in packing the large items as well. Thus the total number of extra bins of capacity 1, denoted by E'_i, is at most $3\epsilon m_i$. Thus all items assigned to bins in B_i have a feasible integral assignment in the set $E'_i \cup B_i$. Now clearly the m_i most profitable bins in the collection $E'_i \cup B_i$ must have a total associated profit of at least $P_i/(1+3\epsilon)$. Moreover, it is easy to verify that all the items in these m_i bins can be packed in the bins of B_i itself.

Small Bin Blocks: Consider now a small bin block B_i. By constraint 3 on the assignment, we know that the profit associated with the violating item in any bin b_j of B_i is at most $\frac{\epsilon}{(1+\epsilon)}p(Y_j)$. Thus we can simply discard all the violating items assigned to bins in B_i and we obtain a feasible solution of profit value at least $P_i/(1+\epsilon)$.

This gives us a feasible integral solution with total profit value at least $\sum_{i\geq 0} P_i/(1+3\epsilon)$. Putting together the guessing and packing steps we obtain our main result.

THEOREM 2.1. *There is a PTAS for the* Multiple Knapsack *problem that provides a $(1+\epsilon)$ approximation in time $n^{O(\ln(1/\epsilon)/\epsilon^8)}$ time.*

3 Generalized Assignment Problem(GAP)

We start by showing that even highly restricted cases of GAP are APX-hard. Next we sketch a 2-approximation algorithm for GAP that easily follows from the work of Shmoys and Tardos [26] on the Min GAP problem.

3.1 APX-hardness of Restricted Instances

We reduce the Maximum 3-bounded 3-Dimensional matching (3DM) problem [6, 14] (defined formally below) in an approximation-preserving manner to highly restricted instances of GAP.

DEFINITION 3.1. (3-BOUNDED 3DM (3DM-3)) *Given a set $T \subseteq X \times Y \times Z$ where $|X| = |Y| = Z| = n$. A matching in T is a subset $M \subseteq T$ such that no elements in M agree in any coordinate. The goal is to find a matching in T of largest cardinality. A 3-bounded instance is one in which the number of occurrences of any element of $X \cup Y \cup Z$ in T is at most 3.*

Kann [14] showed that 3DM-3 is APX-hard, that is, there exists an $\epsilon_0 > 0$ such that it is NP-hard to decide whether an instance has a matching of size n or if every matching has size at most $(1-\epsilon_0)n$. In what follows, we denote by m the number of hyper edges in the set T.

THEOREM 3.1. *GAP is APX-hard even on instances of the following form for all positive δ.*

- $p(i,j) = 1$ *for all items i and bins j.*
- $s(i,j) = 1$ *or* $s(i,j) = 1+\delta$ *for all items i and bins j.*
- $c(j) = 3$ *for all bins j.*

Proof. Given an instance I of 3DM-3 we create an instance $I' = (\mathcal{B}, \mathcal{S})$ of GAP as follows. In I' we have m bins $b_1, \ldots, b_m$ of capacity 3 each, one for each of the edges $e_1, \ldots, e_m$ in T. For each element i of X we have an item x_i in I' and similarly y_j for $j \in Y$ and z_k for $k \in Z$. We also have an additional $2(m-n)$ items in I', $u_1, \ldots, u_{2(m-n)}$. We set all profits to be 1. It remains to set up the sizes. For each item u_h and bin b_ℓ we set $s(u_h, b_\ell) = (1+\delta)$. For an item x_i and bin b_ℓ we set $s(x_i, b_\ell) = 1$ if $i \in e_\ell$ and $(1+\delta)$ otherwise. The sizes of items y_j and z_k are set similarly.

We claim that 3 items can fit in a bin b_ℓ if and only if they are the elements of the edge e_ℓ. Thus bins with 3 items correspond to a matching in T. It then follows that if I has a matching of size n, then I' has a solution of value $3n + 2(m-n)$. Otherwise, every solution to I' has value at most $3n - \epsilon_0 \cdot n + 2(m-n)$. The APX-hardness now follows from the fact that $m = O(n)$ for bounded instances. □

A similar result can be stated if only profits are allowed to vary.

THEOREM 3.2. *GAP is APX-hard even on instances of the following form:*

- *each item takes only two distinct profit values,*
- *each item has an identical size across all bins and there are only two distinct item sizes, and*
- *all bin capacities are identical.*

Proof. The reduction is once again from 3DM-3. Given an instance I of 3DM-3 we create an instance $I' = (\mathcal{B}, \mathcal{S})$ of GAP as follows. In I' we have m bins $b_1, \ldots, b_m$ of capacity 3 each, one for each of the edges $e_1, \ldots, e_m$ in T. For each element i of X we have an item x_i in I' and similarly y_j for $j \in Y$ and z_k for $k \in Z$. We also have an additional $m-n$ items $u_1, ..., u_{m-n}$ where $s(u_h, b_\ell) = 3$ and $p(u_h, b_\ell) = 4$ for any additional item u_h and a bin b_ℓ. Fix a positive constant $\delta < 1/3$. For an item x_i and bin b_ℓ we set $p(x_i, b_\ell) = 1+\delta$ if $i \in e_\ell$ and 1 otherwise. The profits of items y_j and z_k are set similarly. The sizes of items x_i, y_j and z_k are all set to 1 each.

It is now easy to verify that if instance I has a matching of size n, there exists a solution to I' of value $4(m-n) + 3n(1+\delta)$. Otherwise, every solution to I' has value at most

$4(m-n)+3n(1+\delta)-n\epsilon_0 \cdot \delta$. As above, the APX-hardness now follows from the fact that $m = O(n)$. □

Notice that Theorem 3.2 is not a symmetric analogue of Theorem 3.1. In particular, we use items of two different sizes in Theorem 3.2. This is necessary as the special case of GAP where all item sizes are identical across the bins (but the profits can vary from bin to bin), is equivalent to minimum cost bipartite matching.

PROPOSITION 3.1. *There is a polynomial time algorithm to solve GAP instances where all items have identical sizes across the bins.*

3.2 A 2-approximation for GAP

Shmoys and Tardos [26] give a $(1, 2)$ bi-criteria approximation for Min GAP. A paraphrased statement of their precise result is as follows.

THEOREM 3.3. (SHMOYS AND TARDOS [26]) *Given a* feasible *instance for the cost assignment problem, there is a polynomial time algorithm that produces an integral assignment such that*

- *cost of solution is no more than* OPT,
- *each item i assigned to a bin j satisfies $s(i, j) \leq c(j)$, and*
- *if a bin's capacity is violated then there exists a single item that is assigned to the bin whose removal ensures feasibility.*

We now indicate how the above theorem implies a 2-approximation for GAP. The idea is to simply convert the maximization problem to a minimization problem by turning profits into costs by setting $w(i, j) = L - p(i, j)$ where $L > \max_{i,j} p(i, j)$ is a large enough number to make all costs positive. To create a feasible instance we have an additional bin b_{m+1} of capacity 0 and for all items i we set $s(i, m+1) = 0$ and $w(i, m+1) = L$ (in other words $p(i, m+1) = 0$). We then use the algorithm for cost assignment and obtain a solution with the guarantees provided in Theorem 3.3. It is easily seen that the profit obtained by the assignment is at least the optimal profit. Now we show how to obtain a feasible solution of at least half the profit. Let j be any bin whose capacity is violated by the assignment and let i_j be the item guaranteed in Theorem 3.3. If $p(i_j, j)$ is at least half the profit of bin j then we retain i_j and leave out the rest of the items in j. In the other case we leave out i_j. This results in a feasible solution of at least half the profit given by the LP solution. We get the following result:

PROPOSITION 3.2. *There is a 2-approximation for GAP.*

REMARK 3.1. *The algorithm in [26] is based on rounding an LP relaxation. For MKP an optimal solution to the linear program can be easily constructed in $O(n \log n)$ time by first sorting items by their profit to size ratio and then greedily filling them in the bins. Rounding takes $O(n^2 \log n)$ time. We also note that the integrality gap for the LP relaxation of GAP is a factor of 2 even for instances of MKP with identical bin capacities.*

4 A Greedy Algorithm

We now analyse a natural greedy strategy: pack bins one at a time, by applying the FPTAS for the single knapsack problem on the remaining items. Greedy(ϵ) refers to this algorithm with ϵ parameterizing the error tolerance used in the knapsack FPTAS.

CLAIM 4.1. *For instances of MKP with bins of identical capacity Greedy(ϵ) gives a $(\frac{e-1}{e} - O(\epsilon))$-approximation.*

Proof. Let X be the set of items packed by some optimal solution. Let X_j denote the set of items in X that remain after Greedy packs the first $(j-1)$ bins and let Y_j be the items packed by Greedy in the jth bin. Since the bin capacities are identical, by a simple averaging argument it is easy to see that $p(Y_j) \geq (1-\epsilon)p(X_j)/m$. Simple algebra gives the result. □

CLAIM 4.2. *For MKP Greedy(ϵ) gives a $(2+\epsilon)$-approximation.*

Proof. Let X_j denote the set of items that some fixed optimal solution assigns to the jth bin and which do not appear anywhere in Greedy's solution. Also, let Y_j denote the items that Greedy packs in the jth bin. Then we claim that $p(Y_j) \geq (1-\epsilon)p(X_j)$ since X_j was available to be packed when Greedy processed bin j. This follows from the greedy packing. Thus we obtain $\sum_{j=1}^{m} p(Y_j) \geq (1-\epsilon)\sum_{j=1}^{m} p(X_j)$. If $\sum_{j=1}^{m} p(X_j) \geq \mathrm{OPT}/2$ we are done. Otherwise by definition of the X_j's, Greedy must have packed the other half of the profit. This implies the claimed $(2+\epsilon)$-approximation. □

REMARK 4.1. *Claim 4.2 is valid even if the item sizes (but not profits) are a function of the bins, an important special case of GAP that is already APX-hard. The running time of Greedy(ϵ) is $O(mn \log 1/\epsilon + m/\epsilon^4)$ using the algorithm of Lawler [17] for the knapsack problem. Claim 4.2 has been independently observed in [1].*

A Tight Example: We show an instance on which Greedy's performance is no better than 2. There are two items with sizes 1 and $\alpha < 1$ and each has a profit of 1. There are two bins with capacities 1 and α each. Greedy packs the smaller item in the big bin and obtains a profit of 1 while $\text{OPT} = 2$. This also shows that ordering bins in non-increasing capacities does not help improve the performance of Greedy.

5 Conclusions

An interesting aspect of our guessing strategy is that it is completely independent of the number of bins and their capacities. This might prove to be useful in other variants of the knapsack problem. One recent application is in obtaining a PTAS for the stochastic knapsack problem with Bernoulli variables [7].

The Min GAP problem has a $(1, 2)$ bi-criteria approximation and it is NP-hard to obtain a $(1, 3/2 - \epsilon)$-approximation. In contrast GAP has a 2-approximation but the known hardness of approximation is $(1 + \epsilon_0)$ for a very small but fixed ϵ_0. Closing this gap is an interesting open problem.

Another interesting problem is to obtain a PTAS for MKP with an improved running time. Though an FPTAS is ruled out even for the case of two identical bins, a PTAS with a running time of the form $f(1/\epsilon)\text{poly}(n)$ might be achievable. The identical bin capacities case might be more tractable than the general case. Extending our ideas to achieve the above mentioned running time appears to be non-trivial.

References

[1] M. W. Dawande, J. R. Kalagnanam, P. Keskinocak, R. Ravi, and F. S. Salman. Approximation algorithms for the multiple knapsack problem with assignment restrictions. Technical Report, RC21331, IBM T.J. Watson Research Center, 1998.

[2] A. K. Chandra, D. S. Hirschberg, and C. K. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3(3):293–304, Dec 1976.

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

[4] C. E. Ferreira, A. Martin, and R. Weismantel. Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization*, 6(3):858–77, 1996.

[5] A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *European Journal of Operational Research*, 15(1):100–9, 1984.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[7] A. Goel and P. Indyk. Stochastic load balancing and related problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 579–86, 1999.

[8] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

[9] M. S. Hung and J. C. Fisk. An algorithm for 0-1 multiple knapsack problems. *Naval Research Logistical Quarterly*, 24:571–579, 1978.

[10] M. S. Hung and J. C. Fisk. A heuristic routine for solving large loading problems. *Naval Research Logistical Quarterly*, 26(4):643–50, 1979.

[11] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–8, 1975.

[12] G. Ingargiola and J. F. Korsh. An algorithm for the solution of 0-1 loading problems. *Operations Research*, 23(6):110–119, 1975.

[13] J. R. Kalagnanam, M. W. Dawande, M. Trubmo, and H. S. Lee. Inventory problems in steel industry. Technical Report, RC21171, IBM T.J. Watson Research Center, 1998.

[14] V. Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37:27–35, 1991.

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

[16] H. Kellerer. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. In *Proceedings of APPROX'99*, Springer.

[17] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–56, 1979.

[18] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: algorithms and complexity. In S. C. Graves et al. , editor, *Handbooks in OR & MS*, volume 4, pages 445–522. Elsevier Science Publishers, 1993.

[19] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990. Preliminary version appeared in *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, 217–24, 1987.

[20] J. Lin and J. S. Vitter. ϵ-Approximations with minimum packing constraint. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 771–782, 1992.

[21] S. Martello and P. Toth. Solution of the zero-one multiple knapsack problem. *European J. of Operations Research*, 4:322–329, 1980.

[22] S. Martello and P. Toth. A bound and bound algorithm for the zero-one multiple knapsack problem. *Discrete Applied Mathematics*, 3(275–288), 1981.

[23] S. Martello and P. Toth. Heuristics algorithms for the multiple knapsack problem. *Computing*, 27:93–112, 1981.

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

[25] F. D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149–61, 1987.

[26] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–74, 1993. Preliminary version appeared in *Proceedings of the 4th Annual ACM-SIAM Symposium on*

Discrete Algorithms, 448-454, 1993.