LP based heuristics for the multiple knapsack problem with assignment restrictions

Geir Dahl and Njål Foldnes

Centre of Mathematics for Applications and Department of Informatics,

University of Oslo,

P.O.Box 1053 Blindern, 0316 Oslo, Norway

Email: geird, njaalf@ifi.uio.no

Corresp. author: G. Dahl.

Abstract

Starting with a problem in wireless telecommunication, we are led to study the multiple knapsack problem with assignment restrictions. This problem is NP-hard. We consider special cases and their computational complexity. We present both randomized and deterministic LP based algorithms, and show both theoretically and computationally their usefulness for large-scale problems.

Keywords: Multiple knapsack problem, randomized rounding, traffic routing.

Introduction

This paper addresses the *Multiple Knapsack Problem with Assignment Restrictions* (MKARP). In MKARP we are given a set $N = \{1, \dots, n\}$ of items, and a set $M = \{1, \dots, m\}$ of knapsacks. With every item $i \in N$ there are associated a weight $w_i > 0$ and a profit $p_i > 0$, and every knapsack $j \in M$ has a capacity $c_j > 0$. In addition there are assignment restrictions, given by sets $A_i \subseteq M$ for $i \in N$. These sets specify to which knapsacks an item is assignable. In other words, A_i is the set of knapsacks that can hold item i. For each knapsack j we let $B_j \subseteq N$ denote the set of items that are assignable to j. This defines a bipartite graph G with color classes M and N and with the neighbours of $j \in M$ given by A_i. Throughout the paper we assume that $w_i \leq c_j$ whenever item i is assignable to knapsack j.

The presence of assignment restrictions is a generalization of the well known multiple knapsack (MK) problem. MKARP is itself a special case of the generalized assignment problem (for more on this problem and the MK problem, see Martello and Toth (1990) [6]). The situation where $w_i = p_i$ for all items i, was first studied as an independent problem by Dawande et al. (2000) [2]. Furthermore, this problem without assignment restrictions is called the *Multiple Subset Sum Problem* (Caprara, Kellerer and Pferschy (2000) [7]).

Clearly, since MKARP contains MK as a special case, it is NP-hard in the strong sense. In fact, even with the special case where $p_i = w_i$ for all $i \in N$ the problem is NP-hard, as shown in Dawande et al. (2000) [2]. In this paper we first study the computational complexity of various subclasses of MKARP. Since in general MKARP is hard to solve, and the application we have in mind requires fast methods, we then suggest and investigate fast approximation algorithms. Both theoretical and practical

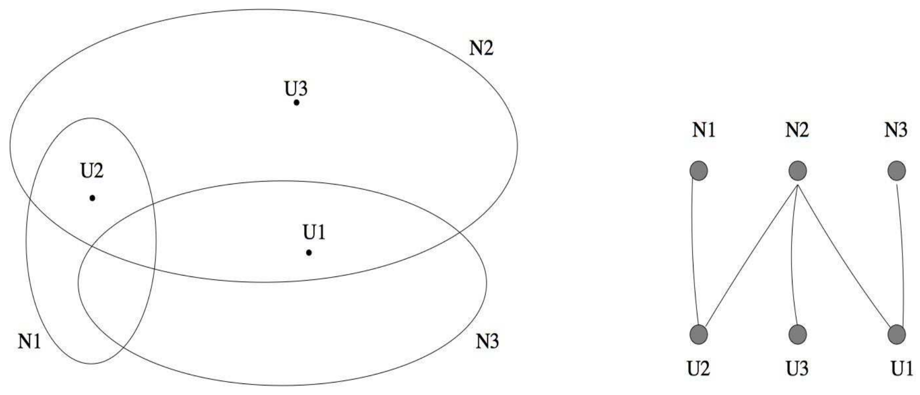

Figure 1: Three users, three antennae.

analysis are given for three linear programming (LP) based approximation algorithms for MKARP.

This study was motivated by the following traffic routing problem in wireless telecommunications. We are given a set of mobile phone users located at different positions in a geographical area. This area is covered by a set of antennae (base stations), each with a certain coverage area. The coverage area of two antennae may overlap, so at each position in the area there may be one, two or more antennae that cover that position (See Fig. , where the three users U1, U2 and U3 are located in a region covered by three networks (antenna) N1, N2 and N3.) Now each mobile user wants a connection, and she may connect to any one of the antennae that cover her location. Associated with each mobile phone user there is a communication demand, and associated with each antenna there is a capacity, i.e., the maximum communication flow it can handle. The goal is to decide for each user whether a connection can be given to her, and, if this is the case, to which antenna she can connect. This must be done so that the total communication flow (given by some reasonable objective function) in the whole region is maximized. MKARP is a natural model for this situation, where we interpete the antennae as knapsacks, and mobile phone users as items. The assignment restrictions can be visualized by the underlying bipartite graph $G = (M \cup N, E)$, where $E = \{\{i, j\} : i \in N, \quad j \in M, \quad j \in A_i\}$, see Fig. .

Now *a feasible assignment* of items to knapsacks is one which respects the capacity constraint for each knapsack and the assignment restriction for each item. In the paper by Dawande et al. (2000) [2] the objective is to maximize the total assigned weight, i.e., the special situation $w_i = p_i$ for all items i. A more general objective is to have a profit p_i for assigning item i (to any knapsack), and the goal is to find a feasible assignment that maximizes the total profit.

Dawande et al. (2000) [2] state that, when $w_i = p_i$ for all items i, MKARP is still NP-hard, and then go on to study approximation algorithms for MKARP. A fast exact method for large multiple knapsack problems, where the ratio m/n is small (typically 0.01), is given in Pisinger (1999) [5]. Ferreira, Martin and Weismantel (1996) [4] present a cutting plane method and heuristics for a slightly more general problem than MKARP, together with some interesting real-life applications.

In this paper, motivated by the application in telecommunications given above, we study some meaningful restrictions on the parameters and the structure of the underlying bipartite graph G. The result is a set of subclasses of MKARP, and we give complexity results for each of these. Then, because of the NP-hardness of MKARP, and the limitation on computational time in the mentioned application (telecommunications), we proceed to study fast approximation methods for the general MKARP. Our heuristics are based on a linear programming relaxation of MKARP, which provide good structural information about optimal solutions. We take profit from this in the design of both deterministic and randomized algorithms. Our algorithms may be seen as alternatives to the 1/2-approximation algorithm presented in Dawande et al. (2000) [2], which is also based on rounding the linear programming solution. However, it is more complicated to implement and there is no computational results. The heuristics we present can handle large instances where the ratio m/n is quite big, with good computational results.

The rest of the paper is as follows. Section 1 treats special cases of MKARP and an analysis of their complexity. Section 2 contains a characterization of the extreme points of the MKARP polyhedron, while section 3 presents the LP based heuristics. Performance bounds are given for the solutions obtained by these methods. Section 4 presents some computational results.

1 Special cases of MKARP

In this section we consider various special cases of MKARP. Especially interesting are special cases that arise from reasonable assumptions with respect to the application in wireless telecommunications. There are basically two types of restrictions we consider. One restriction type is on the values that the parameters p_i, w_i might take. The other is on the structure of the underlying bipartite graph.

The extreme restriction where $m = 1$, i.e., with only one knapsack, reduces to the single knapsack problem. Assuming our goal is to assign maximal weight, we have the optimization version of the SUBSET SUM problem, which is a well-known NP-hard problem, see Garey and Johnson (1979) [10]. So MKARP remains NP-hard even when $m = 1$. Therefore no reasonable restriction on the number of knapsacks will yield a polynomial-time solvable subclass if general item weights and knapsack capacities are allowed.

For the single knapsack problem, it is the unlimited range of weights that cause the problem to be intractable. Moreover, it is natural in the context of telecommunications to allow only a fixed number of possible item weights in the problem. This makes sense since a user may request only a finite number of services (voice, fax, download data, etc). The strongest constraint on the weights is to allow no variation at all, i.e., all weights w_i are identical. This reduces to the bipartite b-matching problem, which is polynomial-time solvable. To be less strict, let us allow the weights to be of two types, the *large* items of weight $q > 1$, and the *small* items of weight 1. Define γ to be the maximum number of large items assigned in any feasible assignment of items. A γ-*solution* is a feasible assignment where γ large items are assigned.

Lemma 1 *Suppose $p_i = w_i$ and that $w_i \in \{1, q\}$ for all items $i \in N$. Then there exists an optimal solution of MKARP that is γ-solution.*

Proof. Consider a feasible assignment with less than γ large items assigned. If we neglect the smaller items, and copy each knapsack j $\lfloor \frac{c_j}{q} \rfloor$ times, we have a bipartite matching problem for the large items. Since the matching of the large items is not maximum, we can find an augmenting path with one exposed leaf in N and the other in M. This gives a new matching of the large items, with one more large item assigned. Only one knapsack, corresponding to the leaf node $j \in M$, has a change in the weight of large items assigned to it. One more large item is assigned to j, so possibly some of the smaller items already assigned to j must be unassigned to obey the capacity constraint at j. But the weight sum of these small items does not exceed q. Therefore we have a new feasible assignment with at least the same total weight assigned, and with one more large item assigned. $\square$

We are now able to show that there are no (non-trivial) restrictions on the values that the weights can take, that keep MKARP polynomially solvable. This is in contrast to the single knapsack problem, where the weights must be exponentially growing in order to obtain exponential running times. In fact, even for the simplest case where $w_i \in \{1, q\}$, and $w_i = p_i$ for all items i, MKARP remains NP-hard.

Proposition 2 *For every fixed $q > 1$, the special case of MKARP where $p_i = w_i$ and $w_i \in \{1, q\}$ for all items $i \in N$ is NP-hard.*

Proof. Our proof is based on transformation from the vertex cover (VC) problem. Recall that an instance of VC is a graph $G = (V, E)$ together with a natural number l. The decision problem is to determine whether there exists an node subset $V' \subseteq V$ such that $|V'| \leq l$, and such that V' covers every edge in E. Given such an instance, we construct the following instance of the $w_j \in \{1, q\}$ MKARP. The knapsacks correspond to the vertices V. Each knapsack has capacity $|E|$. The small items (of weight 1) correspond to E. The small item representing edge $[r, s]$ is admissible to the two knapsack that correspond to nodes r and s. There are $k = |V| - l$ large items of weight $|E|$. A large item is admissible to any knapsack, see Fig. 2. Note that $\gamma = |V| - l$. Now the answer to the VC decision problem is "yes" if and only if MKARP has a γ-solution with all the small items assigned. Also note that from the proof of Lemma 1 it is clear that given an optimal solution, we can construct an optimal solution with γ large items in polynomial time. This implies that there exists a polynomial transformation of VC to the $w_i \in \{1, q\}$ MKARP. $\square$

Hence restrictions based solely on the weight parameters will not lead to polynomial-time solvable problems. However, we might be able to find polynomial-time solvable instances by combining weight restrictions with restrictions on the structure of the

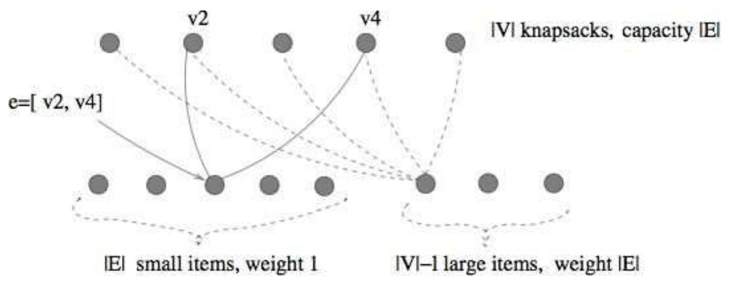
v2
v4
|V| knapsacks, capacity |E|
e=[v2, v4]
|E| small items, weight 1
|V|–1 large items, weight |E|

Figure 2: Proof of Proposition 2

underlying graph. For example, consider the simple problem with only one knapsack and two types of weights. This is clearly polynomial-time solvable.

More generally, let $|w|$ denote the number of possible weight values, and assume that m (the number of knapsacks) and $|w|$ are fixed, i.e., are not part of the problem input. (In telecommunications, this assumption is not unreasonable, as the antennae are the permanent part of the communications system.) The assignment of the items in a weight class may be represented by a vector $x = (x_1, \dots, x_m)$, where x_j is the number of items in this weight class assigned to knapsack j. The number of such vectors x is bounded by $(n+1)^m$. For a given x we find an optimal assignment (in this weight class) by considering items according to decreasing profits. Using such vectors, one for each weight class, we get at most $(n+1)^{m|w|}$ solutions to compare. This gives the following proposition.

Proposition 3 *The special case of MKARP where m and $|w|$ are not part of the input is polynomial-time solvable.*

Another interesting special case is the situation where the cellular phone users are constrained to *move along a road.* That is, the users move along a line, and antennae are placed at various locations along the line. Then the structure of G allows the items to be

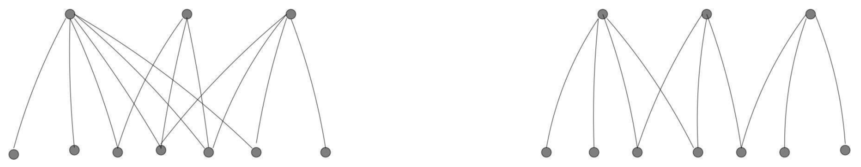

Figure 3: Convex graphs, general and 2-line

ordered in such a way that for each knapsack j, B_j consists of consecutive items. More precisely, for each j there exist integers $l_j \leq r_j$ such that $B_j = \{l_j, l_j + 1, \dots, r_j\}$. Such bipartite graphs are called *convex* (see Glover (1967) [3]). It is not clear whether, in the two-weight situation, this problem is polynomial-time solvable. However, consider the additional restriction that no more than two base stations can cover any location on the road. More precisely for all j we have $l_{j+1} \leq r_j < l_{j+2}$, and consequently the degree of each item node is at most 2. We denote this special case by 2-*line* MKARP, see Fig. 3 for an illustration.

Proposition 4 *The 2-line MKARP with fixed $|w|$ is polynomial-time solvable.*

Proof. We prove this by giving a dynamic programming algorithm for the problem. Assume w.l.o.g. that the underlying graph is connected. For simplicity we assume that $|w| = 2$, i.e., $w_i \in \{q_1, q_2\}$ where $q_1 < q_2$ for all items $i \in N$. The general case $|w| \geq 2$ can be treated similarly. We let I_k denote the items that are assignable to both knapsack k and $k + 1$. Define $f(k, r, s)$ to be the maximum weight that can be assigned to knapsacks $1, \dots, k$, assuming that r small and s large items from I_k is assigned to knapsack k. Furthermore, define $g(k + 1, r, r', s, s')$ to be the maximum weight assignable to knapsack $k + 1$, assuming that r' small and s' large items in I_k are assigned to knapsack k, while r small and s large items in I_{k+1} are assigned to knapsack $k+1$. Then $f(k+1,r,s)$ can be calculated as

$$f(k+1,r,s) = \max_{r',s'}\{f(k,r',s') + q_1 r + s q_2 + g(k+1,r,r',s,s')\}.$$

Here g can calculated in polynomial time, since it is a single knapsack problem with only two weight types involved. Finally, note that calculating $\max_{r,s} f(m,r,s)$ will give the optimal solution in polynomial time.

$\square$

2 Linear relaxation

In this section we study a linear relaxation of MKARP. By using binary variables x_{ij} to denote whether item i is assigned to knapsack j, the following integer program models MKARP :

$$\max \sum_{i\in N}\sum_{j\in A_i} p_i x_{ij}$$

$$\sum_{i\in B_j} w_i x_{ij} \le c_j \qquad j \in M \tag{1}$$

$$\sum_{j\in A_i} x_{ij} \le 1 \qquad i \in N \tag{2}$$

$$x_{ij} \in \{0,1\}. \tag{3}$$

Constraints (1) are the capacity constraints, while constraints (2) ensure that an item can be assigned to at most one knapsack. The *linear relaxation* of MKARP is obtained by relaxing constraint (3) to $0 \le x_{ij} \le 1$. Given an instance I of MKARP, we denote this linear program by LP_I. Let x be any feasible solution to LP_I, and let f denote the subvector of x containing the fractional variables ($0 < x_{ij} < 1$). The *fractional graph* associated with x is the subgraph of the bipartite graph induced by the edges in f. Let P_r denote the polytope defined by (1), (2) and $0 \leq x_{ij} \leq 1$. The following theorem concerning the structure of the extreme points of P_r will be useful.

Theorem 5 *Let x be an extreme point of P_r. Then the following holds:*

> *(i) The fractional graph is a forest, say with component trees $T_1, \ldots, T_\nu$.*
>
> *(ii) Each T_i contains at most one leaf node in N.*
>
> *(iii) Let $i \in \{1, \ldots, \nu\}$. Then all except at most one of the inequalities in (1) and (2) corresponding to the vertices of T_i hold with equality. Moreover, if T_i has a leaf in N, then the corresponding inequality in (2) is strict.*

Proof. Let T be a component of the fractional graph. Suppose that T contains two leaf nodes, say, for simplicity, 1 and l in N. Furthermore we may assume that the path in T between 1 and l (which alternates between vertices in N and M) is defined by the node sequence $1, j_1, 2, j_2 \ldots, l-1, j_{l-1}, l$. Let p be the perturbation vector with subvector corresponding to this path given by $p = (+\epsilon, -\frac{w_1}{w_2}\epsilon, +\frac{w_1}{w_2}\epsilon, -\frac{w_1}{w_3}\epsilon, +\frac{w_1}{w_2}\epsilon, \ldots, +\frac{w_1}{w_{l-1}}\epsilon, -\frac{w_1}{w_l}\epsilon)$, and with zeros for the other arcs. By choosing $\epsilon > 0$ small enough the vectors $v_1 = x + p$ and $v_2 = x - p$ are feasible. In fact, as nodes 1 and l are leaves, the corresponding inequalities in (2) are strict. Moreover, $x = \frac{1}{2}v_1 + \frac{1}{2}v_2$, which contradicts the fact that x is an extreme point. This proves (ii).

To prove (i), suppose the fractional graph contains a cycle. Then by a similar argument, one can find a perturbation of only the variables on the cycle such that x becomes a convex combination of two other points in P_r. This implies that the fractional graph is cycle-free, i.e., it is a tree.

Finally, to prove (iii), note that T_i contains $|V(T)| - 1$ edge variables, all fractional, and since these variables are uniquely determined (vertex property) at least $|V(T)| - 1$

inequalities from (1) and (2) (for vertices in T_i) must be active. Moreover, if T_i has a leaf k, then just one variable x_{kj} ($j \in M$) is positive and fractional, so $\sum_{j \in A_k} x_{kj} < 1$. □

The MKARP heuristics introduced in the next section extract information from an optimal vertex x of P_r. One interesting aspect of x in this respect is the number of variables that are fractional. We call an item or knapsack *fractional* if the corresponding node is the endpoint of some fractional variable in x.

Corollary 6 *Let x be an optimal vertex solution of LP_I. Then the number of fractional items is bounded above by the number of fractional knapsacks.*

Proof. Let T be any tree in the fractional graph, and suppose T has r nodes in M and s nodes in N. The number of edges in T is $r+s-1$. Let θ be the number of leaf nodes in $T \cap N$. From Theorem 5 it follows that $\theta \leq 1$. Since T is a bipartite graph, the sum of the degrees of the nodes in $T \cap N$ is equal to the number of edges. Let d_i denote the degree of node i in T. It follows that $r + s - 1 = \sum_{i \in T \cap N} d_i = \theta + \sum_{i \in T \cap N : d_i \geq 2} d_i \geq \theta + 2(s - \theta)$ from which it follows that $r \geq s$. □

The next corollary will also be used in the following.

Corollary 7 *Let x be an optimal vertex solution of LP_I. Then there exists at least one variable whose value equals 1.*

Proof. If the fractional graph is empty, then x equals 1 in some variable, since we assume that $w_i \leq c_j$ for any pair i, j with $j \in A_i$. So consider a nontrivial tree T in the fractional graph. T contains at least two leaf nodes. By Theorem 5, for at least one of these leaves the corresponding inequality must hold with equality. This leaf must correspond to a knapsack, say $j \in M$. Since j is a leaf, and the corresponding inequality holds with equality, it follows that some item is assigned integrally to j. □

3 LP based heuristics

In this section we present three LP based heuristics for MKARP, together with some theoretical bounds on the quality of the solution values obtained. The computational results for these heuristics are presented in the next section.

The heuristics we present are closely related, in fact they are all iterative rounding schemes. In each iteration the heuristics use information extracted from an optimal vertex solution x of LP_I for some MKARP instance I. We say that an instance I is *trivial* if $A_i = \emptyset$ for all items $i \in N$.

The following is a **Generic scheme** for our heuristics.

1. Initialize: Set I to be the input instance and set $F = \emptyset$.
2. Find an optimal vertex solution x to LP_I.
3. Based on x, assign some of the items. Let A encode this assignment. Update $F \leftarrow F \cup A$.
4. Update I :
    - For each item i and knapsack j such that $[i, j] \in A$:
        - Remove item i from I
        - Set $c_j = c_j - w_i$
    - For each item i in I, if $i \in B_j$ and $w_i > c_j$, set $B_j = B_j \backslash \{i\}$ and $A_i = A_i \backslash \{j\}$.
5. If I is trivial, STOP and output F, else go to Step 2.

Hence, in each iteration we consider an instance I of MKARP. In the first iteration we let I be the original problem. Let A denote the feasible assignment thus obtained ($A \subset N \times M$). Now we "clean up" I in the following way: First we remove from I the items that were assigned in A. Then, for the knapsacks that received these items, the capacities are reduced by subtracting the total weight of assigned items. Finally, due to this capacity reduction, an item that previously were assignable to some knapsack may not be assignable that knapsack any more. This happens when the item weight is greater than the capacity of the knapsack, and the corresponding assignment variables are removed from I. Note that the resulting I is still an instance of MKARP. If I is trivial, we stop, otherwise a new iteration is started.

The following three heuristics, denoted by DET, RAN and COMBI, are distinguished in the way x is used to assign items to knapsacks (Step 3).

The first heuristic is denoted by DET. In this method, only the variables in x that are 1 are considered. More precisely, in DET Step 3 gives the following assignments:

$$A_{\text{DET}} = \{[i,j] : x_{ij} = 1\}.$$

Clearly, A_{DET} is a feasible assignment. Let $v(LP_I)$ denote the optimal value of LP_I. The following proposition provides a lower bound on the solution value $DET(I) = \sum\{w_i : [i,j] \in A_{\text{DET}}\}$ after one iteration of DET.

Theorem 8 *Define $\hat{p}$ to be the total profit of the m most profitable items in N. Let I be an instance of MKARP. Then after the first iteration of DET, the following holds:*

$$DET(I) > v(LP_I) - \hat{p}$$

Proof. By Corollary 6, there are at most m fractional items relative to x. Therefore, by rounding down these fractional values, one looses no more than m items compared to the fractional solution x, whose total profit is at most $\hat{p}$. $\square$

Remark that this result gives a bound on the integrality gap between $v(LP_I)$ and $v(IP)$. In section 4 we shall see that typically this gap is small for large instances. This can be seen by noting that if $\frac{m}{n}$ is small, that is, if there are many items compared with the number of knapsacks, Theorem 8 implies that the solution value obtained by DET tends to be very good, even after the first iteration.

The second heuristic we present is a randomized algorithm denoted by RAN. It is based on randomized rounding (for a survey, see Srinivasan (1999) [8]). In RAN, the initialization (Step 1) also involves the following ordering of the items. The profit/weight-ratio is non-increasing, and if $p_i/w_i = p_j/w_j$ for two items i and j, then $i < j$ if $w_i > w_j$.

RAN follows the generic scheme, where in Step 3 the assignments A_{RAN} are chosen by interpreting the fractional values x_{ij} as probabilities. To be more precise some notation is needed. The probability of an event A is denoted by $P(A)$. We introduce the independent discrete random variables $K_1, K_2, \ldots, K_n$ each with sample space $\{0, 1, \ldots, m\}$ and with probability distributions given by, for each $i \leq n$, $P(K_i = j) = x_{ij}$ $(j = 1, 2, \ldots, m)$ and $P(K_i = 0) = 1 - \sum_{j=1}^{m} x_{ij}$. The interpretation here is that we assign item i to knapsack K_i and this is done with probability x_{ij}. $K_i = 0$ means that i is not assigned to any knapsack. With this notation we obtain a preliminary assignment $\hat{A}_{\mathrm{RAN}}$:

$$\hat{A}_{\mathrm{RAN}} = \{[i, K_i] : i \in N \text{ and } K_i > 0\}.$$

However, this assignment may not be feasible. Consider a knapsack j. Then the total weight assigned to j by $\hat{A}_{\mathrm{RAN}}$ is given by the random variable

$$W^j = \sum_{i=1}^{n} W_i^j$$

where W_i^j is the random variable $W_i^j := w_i \cdot I(K_i = j)$; here $I(K_i = j)$ denotes the indicator function which is 1 in the event that $K_i = j$ and otherwise it is 0. We define $\mu_j = E(W^j)$ and $\sigma_j^2 = Var(W^j)$. Note that $\hat{A}_{\mathrm{RAN}}$ is feasible if and only if $W^j \leq c_j$

for all $j \in M$. If $W^j > c_j$ for knapsack j, we order the set $I_j = \{i \in N : K_i = j\}$ of items assigned to j in non-increasing profit/weight order. Let $b(j)$ be the *break item* of I_j, i.e., the first item in the sequence I_j such that $\sum_{i \in I_j : i \leq b(j)} w_i > c_j$. Then, in order to get a feasible assignment, we define

$$A_{\text{RAN}} = \{[i, K_i] \in \hat{A}_{\text{RAN}} : i < b(K_i)\}.$$

In short, A_{RAN} is obtained by applying a simple greedy algorithm for each knapsack j such that $W^j > c_j$.

It is of interest to understand the properties of the RAN heuristic on a theoretical basis. We therefore give a simplified probabilistic analysis of RAN which seems to explain some of the empirical results that we present in section 4. We restrict the attention to the case when $p_i = w_i$ for all i. Moreover, as our analysis investigates the relation between RAN and the optimal linear relaxation value $v(LP_I) = \sum_j \mu_j$, we may assume that $\mu_j = c_j$ for all j, since this maximizes the gap between $v(LP_I)$ and the value obtained by RAN.

Note that the (preliminary) assigned weight W^j is a sum of independent random variables (as the K_i's are independent). If we knew the probability distribution of W^j it would be possible to calculate, for instance, the probability that the assigned weight W^j does not exceed the capacity c_j. Unfortunately, the probability distribution of W^j is difficult to find (unless the weights w_i are all equal; then we get the binomial distribution). However, an exact bound for W^j can be derived from Proposition 1 in Bertsimas and Vohra (1998) [1]. This results in the following probabilistic tail estimate:

$$P(W^j < (1-\delta)\mu_j) < \exp(-\frac{\delta^2 \mu_j^2}{2 \sum_i w_i^2 x_{ij}}),$$

for every δ with $0 < \delta < 1$. This gives an upper bound on the probability that W^j

Table 1: Values of ρ

n	5	10	20	60	200	600	1000
δ	0.6	.57	0.52	0.40.	0.26	0.17	0.13
ρ	0.12	0.23	0.32	0.51	0.69	0.81	0.85

does not exceed a certain fraction of its expectation. Let the random variable RAN_j denote the weight of the items assigned to j by A_{RAN}. Note that $RAN_j = W^j$ if $W^j \leq c_j$, and that if $W^j > c_j$ one cannot give an explicit expression for RAN_j, other than $RAN_j \geq \mu_j - w_{max}$, where w_{max} is defined by $w_{max} = \max_i w_i$. Clearly, we then have $E(RAN_j) > \min\{(1-\delta)\mu_j, \mu_j - w_{max}\}P(W_j > (1-\delta)\mu_j)$. Using the tail estimate given above, the following bound on the expectation of RAN_j is obtained:

$$E(RAN_j) > \min\{(1-\delta)\mu_j, \mu_j - w_{max}\}(1 - \exp(-\frac{\delta^2\mu_j^2}{2\sum_i w_i^2 x_{ij}})). \tag{4}$$

This makes it possible to obtain a lower bound for the fraction

$$\rho_j = \frac{E(RAN_j)}{\mu_j}$$

for $j \in M$ in any instance I of MKARP. In Table 1 we have calculated the average value of ρ_j for single knapsack problems with various n values. For each n we generate weights and probabilites at random, and in each instance we choose δ such that the right-hand side in (4) is maximized. The values for δ and ρ are average values taken over ten instances.

The values given in Table 1 suggests that the bounds, although not very tight, become better as n increases. It is therefore natural to suggest a different approach for estimating $E(RAN_j)$, and settle with an approximation. The idea is to focus on the situation where m and n are large and use an asymptotic analysis of RAN which we now briefly explain. From probability theory one has the following central limit theorem (which follows directly from Lindeberg's theorem in Billingsley (1995) [11]).

Theorem: *Let $Z_1, Z_2, \ldots$ be a sequence of independent random variables each with expectation zero and satisfying (i) there is a constant C such that $P(|Z_i| \leq C) = 1$ for each i, and (ii) $s_n \to \infty$ as $n \to \infty$ where $s_n^2 = \sum_{i=1}^n Var(Z_i)$. Let $S_n = \sum_{i=1}^n Z_i$. Then S_n/s_n converges in distribution to N, a standard normally distributed variable (i.e., pointwise convergence of the cumulative probability distribution holds).*

We now apply this theorem to a fixed knapsack j and the associated random variables $W_1^j, W_2^j, \ldots, W_n^j$ where n is large. If the weights w_i are uniformly bounded and there is an $\epsilon > 0$ such that $x_{ij} \in [\epsilon, 1-\epsilon]$ for "most" i, then premises of the central limit theorem hold, and we may conclude that the sum $W^j = \sum_{i=1}^n W_i^j$, subtracted by μ_j, and divided by its corresponding standard deviation σ_j, will be approximately (standard) normally distributed, which we simply denote by $(W^j - \mu_j)/\sigma_j \approx N(0,1)$. This opens up for finding an *asymptotic performance guarantee* as indicated in the following theorem, where $RAN(I) = \sum\{w_i : [i, K_i] \in A_{\text{RAN}}\}$ denotes the assigned weight after one iteration of RAN on the instance I.

Theorem 9 *Let I be an instance of MKARP for which $p_i = w_i$ for all items $i \in N$. Assume that the central limit theorem can be applied (see above): $(W^j - \mu_j)/\sigma_j \approx N(0,1)$ for all $j \in M$. Then the following is an approximate lower bound on $E(RAN(I))$:*

$$(1 - \frac{1}{2\alpha} - \frac{1}{\sqrt{2\alpha\pi}}) \cdot v(LP_I),$$

where α is the maximum number such that for any set of α items, and any knapsack, these items will fit in the knapsack.

Proof. Consider a fixed knapsack j, and note again that without loss of generality we may assume that $\mu_j = c_j$. Note that RAN_j is equal to W^j if $W^j \leq c_j$. Otherwise the greedy (by profit/weight) algorithm is applied to j, and $RAN_j \geq (1 - \frac{1}{\alpha})c_j$ in this case, since knapsack j is filled to within $\frac{1}{\alpha}$ of the total capacity after removing the necessary items. Therefore

$$E(RAN_j) \geq \int_0^{\mu_j} zP(dz) + (1 - \frac{1}{\alpha})c_j P(W^j > \mu_j),$$

where P here denotes the probability measure giving the distribution of RAN_j. By standard methods using substitution of variables, the integral above can be shown to be equal to $\mu_j/2 - \sigma_j/\sqrt{2\pi}$, and since $P(W^j > \mu_j) = 0.5$ we get

$$E(RAN_j) \geq (1 - \frac{1}{2\alpha})\mu_j - \frac{\sigma_j}{\sqrt{2\pi}}.$$

Moreover, we may simplify by noting that

$$\sigma_j^2 = \sum_{i \in B_j} w_j^2 x_{ij}(1 - x_{ij}) \leq w_{max}\mu_j \leq \frac{c}{\alpha}\mu_j,$$

which implies that $\sigma_j \leq \mu_j/\sqrt{\alpha}$. The theorem now follows, since $v(LP_I) = \sum_j \mu_j$ and $E(RAN(I)) = \sum_j E(RAN_j)$. □

As an example, consider an instance I in which any knapsack j can hold any set of 20 items from B_j. If the premises of Theorem 9 hold, we obtain the bound $RAN(I) \geq 0.97 \cdot LP_I$ by setting $\alpha = 20$. So asymptotically, RAN has a very good expected value.

The last heuristic we consider, denoted by COMBI, is a combination of the previous two heuristics. In COMBI, Step 3 is carried out by combining DET and RAN in the following way. COMBI starts by performing Step 3 of DET, i.e., by first assigning all items i for which there is knapsack j such that $x_{ij} = 1$. Then Step 3 of RAN is carried out, by running through the unassigned items with randomized rounding, using the fractional values of x. So COMBI is a randomized iterative method that in each iteration first assigns the same items as DET does, and then randomly assigns the other items. Therefore the contribution of COMBI in each iteration is at least that of DET. We do not describe Step 3 in detail for this heuristic.

4 Computational experiments

We briefly report some computational experience with the algorithms developed in this paper. The algorithms were implemented in C++ and run on a 1015 MHz Sun Sparc machine. For the solution of linear and integer programs we used CPLEX version 7.5 [9].

The instances were randomly generated. Item weights are uniformly distributed in the interval $[10, R]$ for various values of R. We consider three types of instances, based on the relation between the w_i and the p_i. In the first type, denoted by A, $p_i = w_i$ for all items i. In the second type, B, the profits p_i are uniformly distributed in $[10, R]$, while in the third type, C, $p_i = w_i + 10$ for all i. In each problem the capacities c_j are uniformly distributed in intervals that are chosen such that the total capacity of the knapsacks is roughly equal to half of the total weight of the items.

The following list explains the abbreviations used in the tables in this section:

# Vars:	Number of variables
Tot w:	Total weight of items
Tot c:	Total weight of knapsacks
Type:	Type of problem
R:	Range of weight and profit values
IP time:	Integer program cpu time
#BB:	Number of branch & bound nodes
$v(IP)$:	Optimal value of IP
$v(LP_I)$:	Optimal value of LP_I
# iter:	The number of iterations
value:	The solution value (given in %)

The main conclusions drawn from our computational experiments are :

- In general, for large instances (α is large and/or n is large) it is easy to find good approximate solutions. All heuristics we tested performed well. As the problem size grows, MKARP practically becomes a continuous problem, and as α is very big for these instances, each item does not contribute much individually.
- The integrality gap (the difference between $v(LP_I)$ and $v(IP)$) for most instances is almost tight typically within 2-5%. This is also confirmed theoretically by Theorem 8. It is surprising, however, that the branch and bound tree is still very large. So even rather small instances may be difficult to solve to optimality using CPLEX.
- The LP based heuristics are robust, and perform generally well. RAN seems to give the best results. For some types of instances, the LP based heuristics are substantially better than a simple greedy algorithm GR (explained below), see Table 2.

The simple algorithm GR mentioned in the last point above, is the following: Consider the items in non-increasing profit/weight ratio, and assign item i to the first (if any) knapsack $j \in A_i$ with sufficient rest-capacity to hold i. In Table 2 we compare GR, DET and RAN on 13 instances. Note that they all have rather sparse underlying graphs. We see that the LP based heuristics, especially RAN, outperform GR. COMBI is not included in our table, as it gives values similar to RAN and DET.

Table 2: GR, DET and RAN comparison.

Problem	m	n	#Vars	GR	DET	RAN
sp1	4	20	33	94.8	97.4	97.4
sp2	4	20	30	82.3	88.6	91.0
sp3	4	20	32	84.5	94.4	94.4
sp4	3	10	21	78.1	100	100
sp5	3	10	18	86.2	94.8	100
mp1	100	100	311	86.6	100	100
mp2	100	100	306	90.4	94.7	94.7
mp3	200	400	963	91.7	93.3	94.0
mp4	400	400	1041	85.7	100	100
mp5	1000	1000	1737	93.1	100	100
mp6	300	500	1004	91.9	99.6	95.4
mp7	50	100	138	93.2	95.2	95.2
mp8	50	100	207	90.7	95.1	95.9

Table 3 gives a summary of 9 instances of size $m = 5$ and $n = 50$. For the randomized algorithms RAN and COMBI, the reported solution value is the best after running the algorithm five times on each instance. To solve LP_I using CPLEX, note that we used the primal simplex method (without preprocessing) as, for these problems, this method is much faster than the dual simplex method. Some of the instances in Table 3 could not be solved to optimality by CPLEX; this is denoted by '-' in the table. As mentioned in our main conclusion this is perhaps surprising due to the excellent LP bound and rather small problem size.

Table 3: Problems p1-p9, $m = 5$ and $n = 50$.

Problem	# Vars	Tot w	Tot c	Type	R	ip time	#BB	$v(IP)$	$v(LP_I)$
p1	167	743	453	A	10	0.26	231	453	453.0
p2	109	741	387	B	10	2.1	1545	540	540.6
p3	149	738	540	C	10	-	-	-	935.9
p4	141	2853	1768	A	100	3.6	5957	1768	1768.0
p5	94	2732	1296	B	100	0.5	211	2125	2153.6
p6	145	2713	1587	C	100	41.14	53337	1957	1959.6
p7	156	20191	15950	A	1000	-	-	-	15950
p8	120	27817	18269	B	1000	90	134871	24101	24142.0
p9	94	24599	12892	C	1000	-	-	-	13253.0

Table 4 reports the computational results of running our three heuristics and GR on the same instances as given in Table 3 . The solution value, in column 'value', is given with respect to $v(IP)$ if this is available. Otherwise the solution value is compared to $v(LP_I)$, and this is indicated with a '*' attached to the problem name. We see that GR perform at least as well as RAN on type A problems. However, for instances of type B and, especially, of type C, RAN gives better solution values than GR.

Table 4: Computational results, problems p1-p9

	DET			RAN			COMBI			GR
Problem	time	# iter	value	time	# iter	value	time	# iter	value	value
p1	0	3	96.3	0	2	97.4	0	2	96.3	97.6
p2	0	3	95.9	0	2	96.1	0	2	96.6	93.5
p3*	0	2	92.7	0.01	2	94.7	0	1	92.7	92.8
p4	0	3	98.2	0	3	99.8	0	2	99.7	99.7
p5	0	2	96.0	0.01	2	96.7	0.01	2	96.7	94.7
p6	0	2	91.4	0.01	2	97.5	0	1	92.7	91.4
p7*	0.01	3	99.6	0.01	3	99.8	0.01	4	99.7	99.6
p8	0	3	97.1	0	2	97.1	0	2	97.1	95.9
p9*	0.01	2	89.5	0	1	96.7	0	1	89.9	92.3

Experiments for problems of larger size, with $m = 100$ and $n = 1000$ were also performed. The LP based heuristics were able to solve these in under a second. Due to the large size, as mentioned in our main conclusion, all heuristics performed very well on these instances. In general it might be sufficient to use GR for problems of this size, where α and n are large.

We also considered some very large instances, with $m = 200$ and $n = 4000$, and around 250 000 variables. For such instances solving the LP can take tens of seconds, making up the main part or the total running time. To reduce this time one might construct faster specialized algorithms for the linear program, based on network flows and some post-processing to ensure the extreme point property of x (see Dawande et al. (2000) [2]).

Acknowledgment. The authors thank the referee for his/her useful comments and suggestions.

References

[1] D. Bertsimas and R. Vohra. (1998). Rounding algorithms for covering problems *Mathematical Programming*, Volume 80, issue 1, page 63.

[2] M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi and F.S. Salman. (2000). Approximation algorithms for the multiple knapsack problem with assignment restrictions, *Journal of Combinatorial Optimization 4*, 171-186.

[3] F. Glover. (1967) Maximum matching in a convex bipartite graph, *Naval research logistics quarterly.*

[4] C.E. Ferreira, A. Martin and R. Weismantel. (1996). Solving multiple knapsack problems by cutting planes *SIAM J. Optimization*, Vol.6, No.3, pp. 858-877.

[5] D. Pisinger. (1999). An exact algorithm for large multiple knapsack problems *European Journal of Operational Research,* 114 pp.528-541

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

[7] A. Caprara, H. Kellerer and U. Pferschy. (2000). A PTAS for the Multiple Subset Sum Problem with different knapsack capacities, *Information Processing Letters 73*, 111-118.

[8] A. Srinivasan. (1999). Approximation algorithms via randomized rounding: A survey, *Lectures on Approximation and Randomized Algorithms (M. Karonski and H. J. Promel, editors), Series in Advanced Topics in Mathematics*, Polish Scientific Publishers PWN, Warszawa, pages 9-71.

[9] CPLEX Optimization, Inc.

[10] M.R. Garey and D.S. Johnson(1979). *Computers and Intractability: A Guide to NP-Completeness.* WH Freeman.

[11] P. Billingsley. (1995). *Probability and Measure.* New York: Wiley.