A Greedy Approximation Algorithm for the Multi-dimensional Minimum Knapsack Problem

Christophe Rapine\textsuperscript{a,}\textsuperscript{*}, Hande Yaman\textsuperscript{b}

\textsuperscript{a}Laboratory G-SCOP / CNRS, Grenoble INP, 46 av. Félix-Viallet, Grenoble, France
\textsuperscript{b}Bilkent University, Department of Industrial Engineering, 06800 Bilkent, Ankara, Turkey

Abstract

We consider the minimization version of the multi-dimensional knapsack problem with bounded integer variables. We propose a greedy approximation algorithm that runs in $O(mn^2)$ time and has a guarantee of $m + 1$, where $n$ is the number of items and $m$ is the number of knapsack constraints. The guarantee of the algorithm can be improved in case of easy coefficients in the constraints associated with some of the dimensions. As a consequence, we show that the greedy is a 2-approximation algorithm for the multiple choice minimization knapsack problem. Finally, using filtering based on the greedy algorithm and LP rounding, we improve the guarantee bounds.

Key words: Multi-dimensional Knapsack, Minimization Knapsack, Approximation, Greedy algorithm

1. Introduction

In this study, we consider the minimization version of the multi-dimensional knapsack problem (mD-minKP). We are given a set of items $N = \{1, \ldots, n\}$, where each item $j$ has cost $f_j$ and is available in $b_j$ copies. With each item $j \in N$, we associate a vector of nonnegative utilities $(u_{1j}, \ldots, u_{mj})$. The problem is to decide on how many copies of each item to select so that the overall utility is at least a specified amount $D_i$ in each dimension $i = 1, \ldots, m$ and the total cost is minimized. The mD-minKP can be modeled by the following integer formulation $IP$:

\[
\begin{align*}
\text{min} & \quad \sum_{j \in N} f_j x_j \\
\text{s.t.} & \quad \sum_{j \in N} u_{ij} x_j \geq D_i \quad \forall i = 1, \ldots, m, \\
& \quad x_j \leq b_j \quad \forall j \in N, \\
& \quad x_j \in \mathbb{Z}_+ \quad \forall j \in N,
\end{align*}
\]

where variable $x_j$ is the number of selected copies of item $j \in N$. We assume without loss of generality that cost $f_j > 0$ for all items $j \in N$ and $D_i > 0$ for all $i = 1, \ldots, m$. The one dimensional case where $b_j = 1$ for all $j \in N$ is called the binary minimization knapsack problem ($\{0,1\}$-minKP), while the one with $b_j = +\infty$ for all $j \in N$ is referred to as the unbounded (integer) minimization knapsack problem. This minimization version of the knapsack problem is also referred in literature as the covering problem.
problem, while the maximization version is called the packing problem. All these problems are well known to be \( \mathcal{NP} \)-hard, by reduction from the classical \( \{0,1\} \) maximization knapsack problem. Notice that if \( m \) is not fixed, i.e., if it is part of the instance, formulation \( IP \) is a general integer linear program with all coefficients nonnegative. Also observe that minimum set cover is a particular case of the binary minimization knapsack problem, corresponding to all coefficients being either 0 or 1, where each subset is viewed as an item whose utility vector is its incident vector and each element results in a unit demand to satisfy.

The aim of this paper is to propose a new approximation algorithm based on a greedy approach for problem \( mD\text{-}\text{minKP} \). Recall that a polynomial time algorithm for a minimization problem is said to be a \( \lambda \)-approximation, or to have a performance guarantee of \( \lambda \), if for any instance it delivers a solution of cost no more than \( \lambda \) times the minimum cost. If \( m \) is not fixed, as minimum set cover is a particular case of \( mD\text{-}\text{minKP} \), it is unlikely to approximate it within a guarantee \( O(\log m) \), unless \( \mathcal{NP} = \mathcal{NP} \) (see [15]).

Even though there is an extensive literature on maximization knapsack problems (see for instance the books by Martello and Toth [12] and Kellerer et al. [10]), the studies on the minimization version are few. If all upper bounds are finite, \( mD\text{-}\text{minKP} \) can be converted into a maximization knapsack problem by complementing the variables, i.e., by letting \( \hat{x}_j = b_j - x_j \) for all \( j \in N \). Hence any exact algorithm for the maximization version can be used to solve the minimization version. This explains why there are fewer studies on the minimization knapsack problems. However, the approach of complementing variables and posing the problem as a maximization problem cannot be used in general when we investigate the performance guarantee of approximation algorithms. This difference is the main motivation of the current study.

Below, we summarize the studies on approximation algorithms for the minimization knapsack problems. All the studies that we have encountered are on the one dimensional case (\( m = 1 \)), often restricted to the \( \{0,1\}\text{-}\text{minKP} \) problem. A paper closely related to our work is due to Csirik et al. [4]. The authors mention previous publications on this problem translated from studies in Russian. They analyze a greedy heuristic proposed by Gens and Levner [8] for the \( \{0,1\}\text{-}\text{minKP} \) and establish that it has a performance guarantee of 2. Then they refine the heuristic to provide an improved guarantee of \( 3/2 \) and mention that it is possible to develop an \((1 + \epsilon)\)-approximation scheme (PTAS) using the idea of the refinement. Güntzer and Jungnickel [9] propose a similar greedy algorithm, still for the \( \{0,1\}\text{-}\text{minKP} \). They prove that this algorithm has a guarantee of 2. They present several approximation algorithms with improved bounds for the special case of the subset sum problem (\( u_{ij} = f_j \) for all items \( j \in N \)). More recently, Carnes and Shmoys [3] present a primal-dual algorithm for the \( \{0,1\}\text{-}\text{minKP} \) with the same guarantee of 2. Their algorithm is based on a formulation that uses flow cover inequalities. The authors extend the algorithm to the single demand facility location and single item lot sizing problems.

Labbé et al. [11] study the one dimensional \( \text{minKP} \) with bounded integer variables, consider-
ing unit utilities but arbitrary nondecreasing cost functions. They present a fully polynomial approximation scheme (FPTAS) for the general case and a polynomial time heuristic algorithm for the case of concave and piecewise linear cost functions. The performance guarantee of this heuristic is 2 when the cost function is linear for positive values of variables and zero otherwise.

Finally, we note that even though there are many studies on approximation algorithms on the maximization version of the multi-dimensional 0-1 knapsack problem (see [7] for a survey), few studies exist on the multi-dimensional problem with bounded integer variables. A common heuristic approach is to define for each item \( j \) a pseudo-utility as a convex combination \( \mu_j = \sum_{i=1}^{m} \lambda_i u_{ij} \) of its utilities, and to apply some greedy algorithms based on these pseudo-utilities. This corresponds to considering the surrogate relaxation of the problem, resulting in a one dimensional knapsack problem. However, choosing "good" multipliers \( \lambda_i \) for a greedy approach to perform well is not trivial, and may be in fact an \( \mathcal{NP} \)-hard problem [16]. As an alternative, Akçay et al. [1], in one of the few studies considering the bounded integer knapsack problem, propose to select the items based on their effective utility. They argue that this criterion, which is imposed by a single bottleneck constraint for each item, is expected to reflect better the importance of an item than its pseudo-utility, which aggregates all the resources. In this paper, like [1], we do not consider the pseudo-utility of an item, but only its actual utility with respect to each dimension.

To the best of our knowledge, there is no previous study on the multi-dimensional minimization knapsack problem with bounded integer variables. This paper contributes to the existing literature by proposing a greedy approximation algorithm for mD-minKP. The algorithm is based on a dynamic programming recursion and has a time complexity of \( \mathcal{O}(mn^2) \) and a guarantee of \( m + 1 \). The algorithm is easy and has the flexibility of being combined with different algorithms in its stages. More precisely, as the guarantee is computed recursively, if there are easy constraints, i.e., constraints over which the optimization problem is easy, then the guarantee can be improved by solving the associated knapsack problems optimally. As a consequence, the algorithm has a guarantee \( m + 1 \) for the multi-dimensional problem with \( m \) general knapsack constraints and a set of cardinality constraints over which it is easy to optimize, such as multiple choice constraints or a laminar family of cardinality constraints. The guarantee can also be improved for other easy cases such as divisible or cross ratio ordered coefficients. We further suggest ways to improve the guarantee on general instances using filtering together with greedy and LP rounding algorithms.

In Section 2, we review the greedy algorithm for the special case of \( \{0,1\}-\text{minKP} \) and give a simpler proof of its guarantee. In Section 3, we generalize our algorithm to the case with arbitrary upper bounds and \( m \) dimensions. We discuss how to improve the guarantee of the algorithm for easy knapsack constraints in Section 4. In Section 5, we combine filtering with the greedy algorithm and LP rounding. We conclude the paper in Section 6.
2. A greedy algorithm for \(\{0,1\}\)-minKP

In this section, we present a greedy approximation algorithm for \(\{0,1\}\)-minKP. Since we have a single dimension in the \(\{0,1\}\)-minKP, we drop the index of the dimension: for short, the utility of item \(j \in N\) is denoted by \(u_j\) and the knapsack requirement is \(D\). The algorithm is based on the density \(d_j\) of the items, defined for each item \(j\) as the ratio \(f_j/u_j\). In the sequel, we assume that the items are indexed in non-decreasing order of their densities, which can done in a \(O(n \log n)\) preprocessing step.

The algorithm then runs in linear time and gives a feasible solution whose cost is at most two times the optimal cost. Algorithm Greedy is identical to the one proposed by Gens and Levner [8] and studied by Csirik et al. [4]. We present it to illustrate the basic definitions and the ideas that we later use for the mD-minKP. In addition, we provide a simpler proof of its performance guarantee.

Algorithm Greedy starts with an empty set \(G\) of selected items. Then iteratively it adds to \(G\) the remaining item \(l\) of smallest density. However if it happens that \(S = G \cup \{l\}\) is feasible, that is, if the total utility is greater than or equal to \(D\), solution \(S\) is memorized if it is better than the best solution found and item \(l\) is discarded (not included in \(G\)). The algorithm stops when no item remains, i.e., when all items are either in \(G\) or have been discarded, and returns the best feasible solution found in its course. For an item \(l \in N\), we define \(G_l = G \cap \{1, \ldots, l\}\) as the greedy set built by the algorithm on items \(\{1, \ldots, l\}\). Notice that this set includes item \(l\) only if it is not discarded.

To analyze the performance of Greedy, we consider in the following an optimal solution \(S^*\) of cost \(OPT\). Observe that by construction the total utility of \(G_n\) is smaller than \(D\). It follows that \(S^*\) contains at least one discarded item. Let \(l^*\) be the discarded item of smallest index belonging to \(S^*\).

We can make the following observation:

**Lemma 1.** \(f(G_{l^*})\) is a lower bound of \(OPT\).

**Proof.** Let \(M\) be the set of all items other than the discarded items of index strictly smaller than \(l^*\), that is \(M ≡ G_{l^*} \cup \{l^*, \ldots, n\}\). Notice that the optimum value of the instance restricted to set \(M\) is also \(OPT\), since \(S^* \subseteq M\). By definition, set \(G_{l^*}\) is exactly the set of the \(|G_{l^*}|\) items of smallest density of \(M\). Since \(u(G_{l^*}) < D\), \(f(G_{l^*})\) is a lower bound on \(OPT\). \(\square\)

We can now derive the following theorem establishing the performance guarantee of Greedy:

**Theorem 1.** Greedy is a 2-approximation, running in linear time \(O(n)\) if the items are sorted in non-decreasing order of the densities.

**Proof.** Consider the solution \(S = G_{l^*} \cup \{l^*\}\). Since \(l^*\) is discarded, by definition \(u(G_{l^*}) + u_{l^*} \geq D\) and hence \(S\) is feasible. The cost of \(S\) is \(f(S) = f(G_{l^*}) + f_{l^*}\). Using Lemma 1 and the fact that \(l^*\) belongs to \(S^*\), we can bound each term by \(OPT\) and conclude that \(f(S) \leq 2OPT\). Finally, it can be readily seen that \(S\) is one of the feasible solutions considered by the algorithm. This implies that Greedy returns a solution of cost at most \(f(S) \leq 2OPT\). It is easy to see that this algorithm runs in linear time. \(\square\)
3. A greedy-based algorithm for mD-minKP

In this section we generalize algorithm Greedy for \{0,1\}-minKP, both by considering any dimension \(m\) and a finite integer upper bound \(b_j\) for each item \(j \in N\). As a solution is then a multiset, we represent it with its multiplicity vector \(x\) used in formulation \(IP\), i.e., \(x_j\) is the number of copies of item \(j\) in the solution. In the following, we denote by \(u_i(x) \equiv \sum_{j \in N} u_{ij} x_j\) the utility of solution \(x\) related to the \(i\)th dimension for \(i = 1, \ldots, m\).

The principle of the algorithm is to apply the greedy algorithm on a single knapsack relaxation, considering for instance only the last constraint with requirement \(D_m\). As in the one dimensional case, for each greedy solution built for this relaxation, we extend it to a feasible solution that satisfies all the \(m\) knapsack constraints. In doing this, each time we encounter a discarded item \(l\), we account it at the cost of a feasible solution for the \((m-1)\) remaining constraints where we impose item \(l\) to be selected. Below, we explain this procedure in more details.

For an index \(i \in \{1, \ldots, m\}\) and an item \(k \in N\), let \(IP(i, k)\) be the \(i\)-dimensional minimum knapsack problem with the knapsack constraints 1 to \(i\) and the additional constraint that \(x_k \geq 1\). In other words, \(IP(i, k)\) is the problem in which we relax the last \((m-i)\) constraints of formulation \(IP\) while imposing that the solution selects at least one copy of item \(k\). We denote by OPT\((i, k)\) the optimal value of problem \(IP(i, k)\). Observe that if some optimal solution of \(IP\) contains at least one copy of item \(k\), then OPT\((i, k)\) is a lower bound of OPT. For convenience we also introduce problem \(IP(0, k)\), where the only constraint is to use one copy of item \(k\). Obviously we have OPT\((0, k) = f_k\).

Our approach builds a feasible solution \(\hat{x}(i, k)\) for problem \(IP(i, k)\) based on the greedy algorithm presented in the previous section and the recursive knowledge of a feasible solution \(\hat{x}(i-1, j)\) for each possible item \(j\). In the case \(i = 0\), solution \(\hat{x}(0, k)\) consists in selecting one copy of item \(k\). For index \(i > 1\), as in the one dimension case, we consider a series of solutions \(x\) by adding greedily all the copies of each item, and discarding the items making \(x\) satisfy demand \(D_i\). The greedy selection is based on the density \(d_{ij}\) of item \(j\) related to the \(i\)th knapsack, defined as \(d_{ij} = f_j/u_{ij}\). Initially, solution \(x\) of selected items contains a single copy of item \(k\) to fulfill the specific constraint of \(IP(i, k)\).

At each step, we consider the item, say \(l\), with the smallest density among the remaining items. If the total utility of the current solution \(x\) and the \(b_l\) copies of \(l\) does not satisfy demand \(D_i\), then we select all the copies of \(l\) by setting \(x_l = b_l\). In this case item \(l\) is said to be fully-copied. Otherwise, item \(l\) is discarded and a feasible solution \(y(l)\) that uses this item is considered as a candidate. Solution \(y(l)\) is constructed by adding sufficiently many copies of item \(l\) to \(x\) to satisfy demand \(D_i\) and by completing it with a feasible solution \(\hat{x}(i-1, l)\) for problem \(IP(i-1, l)\). More precisely, we do the following steps:

- Let \(a_l\) be the maximum number of copies of item \(l\) such that when added to \(x\), the demand \(D_i\) is not satisfied, i.e., \(a_l \equiv \left\lceil (D_i - u_i(x))/u_{il} \right\rceil - 1\). Since \(l\) can not be fully-copied, we have \(0 \leq a_l < b_l\). We de-
note by \(x^{(l)}\) the solution obtained by adding \(a_l\) copies of \(l\) to the current solution \(x\), that is \(x_j^{(l)} = x_j\) for all item \(j \in N \setminus \{l\}\) and \(x_l^{(l)} = a_l\). Note that, by construction, we have \(u_i(x^{(l)}) < D_i \leq u_i(x^{(l)}) + u_{il}\).

- Solution \(y^{(l)}\) is constructed by taking \(y_j^{(l)} = \min\{x_j^{(l)} + \hat{x}_j^{(i-1,l)}, b_j\}\) for each item \(j \in N\).

The algorithm terminates when all items have been considered. We set \(\hat{x}^{(i,k)}\) to the best solution \(y^{(l)}\) over all discarded items \(l\). Lemma 2 states that \(\hat{x}^{(i,l)}\) is a feasible solution for \(IP(i,k)\).

**Lemma 2.** For each discarded item \(l\), \(y^{(l)}\) is a feasible solution for \(IP(i,k)\).

**Proof.** First observe that by construction \(y_k^{(l)} \geq 1\), i.e., the constraint of selecting at least one copy of \(k\) is satisfied. Since \(y^{(l)} \geq \hat{x}^{(i-1,l)}\), \(y^{(l)}\) also satisfies the first \((i-1)\) knapsack constraints. Finally, for all \(j \in N \setminus \{l\}\) we have \(y_j^{(l)} \geq x_j\) and, since \(\hat{x}_i^{(i-1,l)} \geq 1\) and \(b_l > a_l\), we also have \(y_l^{(l)} \geq a_l + 1\). It implies that the \(i\)th knapsack constraint is also satisfied. □

Our approach is based on dynamic programming. However memorizing solution \(\hat{x}^{(i-1,l)}\) for each subproblem and computing the exact cost of \(y^{(l)}\) requires memory and time. Indeed, evaluating the cost of a solution requires \(O(n)\) operations. Instead, our algorithm \(m\)-GREEDY computes an upper bound \(\mathcal{C}(i,k)\) on the actual cost of \(\hat{x}^{(i,k)}\), and thus an upper bound on the optimum value \(OPT(i,k)\). For this, we use the fact that \(f(y^{(l)}) \leq f(x^{(l)}) + f(\hat{x}^{(i-1,l)})\). Assuming that we know an upper bound \(\mathcal{C}(i-1,l)\) of \(f(\hat{x}^{(i-1,l)})\), we can over-estimate \(f(y^{(l)})\) with the quantity \(f(x^{(l)}) + \mathcal{C}(i-1,l)\). Observe that this corresponds to completing solution \(x^{(l)}\) with one more copy of item \(l\), accounted for cost \(\mathcal{C}(i-1,l)\), to satisfy demand \(D_i\). The algorithm returns \(\mathcal{C}(i,k)\), the smallest upper bound computed in its course

\[
\mathcal{C}(i,k) = \min\{f(x^{(l)}) + \mathcal{C}(i-1,l) \mid l \text{ discarded}\}
\]

where we use the zero dimensional knapsack as the basis of the recursion with \(\mathcal{C}(i,k) = f_k\) for all \(k \in N\). In the special case where \(u_{ik} \geq D_i\), i.e., a single copy of item \(k\) satisfies the \(i\)th constraint, we define \(\mathcal{C}(i,k)\) simply as \(\mathcal{C}(i-1,k)\). A description of the computation of \(\mathcal{C}(i,k)\) is given in Algorithm 1.

As in the one dimensional case, we establish the guarantee of our approach by deriving two lower bounds on the optimal value. Let \(x^*\) be an optimal solution for problem \(IP(i,k)\), of cost \(OPT(i,k)\). By definition, we have \(x^*_x \geq 1\). Notice that by construction of the greedy, if \(u_{ik} < D_i\), we have \(u_i(x) < D_i\) at each iteration. Thus there exists at least one discarded item \(l\) such that \(x^*_l \geq 1\). Let \(l^*\) be the discarded item of smallest index belonging to \(x^*\) and let \(x^{(l^*)}\) be the corresponding greedy vector. In the case where \(u_{ik} \geq D_i\), we define by convention \(l^*\) to be \(k\) and \(x^{(l^*)}\) to be the null vector. We have the following lower bound:

**Lemma 3.** \(f(x^{(l^*)})\) is a lower bound of \(OPT(i,k)\).

**Proof.** As before, define \(M\) to be the set of all the items except the discarded items of index strictly smaller than \(l^*\). The optimum value of the instance of \(IP(i,k)\) restricted to set \(M\) is also \(OPT(i,k)\), as \(x^*_j \geq 1\) implies that item \(j\) is in \(M\) by our choice of \(l^*\). In addition, except possibly for item \(k\), in solution \(x^{(l^*)}\) the items of \(M\) of smallest density are
all fully-copied, till item \( l^* \). It results that any solution of utility greater than \( u_i(x^{l^*}) \) costs at least \( f(x^{l^*}) \). Since \( x^{l^*} \) does not satisfy demand \( D_i \), the result follows.

In addition, since \( l^* \) belongs to the optimal solution \( x^* \), we have a second immediate lower bound:

**Lemma 4.** \( \text{OPT}(i-1, l^*) \leq \text{OPT}(i, k) \).

**Proof.** Since \( x^*_k \) and \( x^*_l \) are both not zero, we have \( f(x^*) = \text{OPT}(i, k) = \text{OPT}(i, l^*) \). As problem \( IP(i-1, l^*) \) is a relaxation of problem \( IP(i, l^*) \), the result follows.

Using these two lower bounds, we prove a recursive guarantee result:

**Theorem 2.** Let \( \alpha \) be a value such that \( \mathcal{C}(i-1, l) \leq \alpha \text{OPT}(i-1, l) \) holds for all items \( l \in N \). Then \( \mathcal{C}(i, k) \leq (\alpha + 1)\text{OPT}(i, k) \) for any item \( k \).

**Proof.** Let \( l^* \) be the first discarded item of an optimal solution for \( IP(i, k) \). As \( l^* \) is discarded, we have \( \mathcal{C}(i, k) \leq f(x^{l^*}) + \mathcal{C}(i-1, l^*) \). By Lemma 3, we know that \( f(x^{l^*}) \leq \text{OPT}(i, k) \). In addition, \( \mathcal{C}(i-1, l^*) \leq \alpha \text{OPT}(i-1, l^*) \) and by Lemma 4, we have \( \text{OPT}(i-1, l^*) \leq \text{OPT}(i, k) \). Hence, \( \mathcal{C}(i, k) \leq (\alpha + 1)\text{OPT}(i, k) \).

Theorem 2 says that each additional knapsack constraint deteriorates the guarantee of our greedy algorithm by an additive term of 1. We call \( m \)-\text{Greedy}, the algorithm using dynamic programming to compute costs \( \mathcal{C}(i, k) \) for all \( i \) and \( k \), and returning the minimal cost \( \mathcal{C}(m, k) \) over all items \( k \in N \). Notice that we have \( \mathcal{C}(0, k) = \text{OPT}(0, k) \) for any item \( k \in N \). Thus as a corollary of Theorem 2, we obtain the following result:

**Corollary 1.** Algorithm \( m \)-\text{Greedy} is a \((m+1)\)-approximation for the \( mD\text{-minKP} \), with time complexity \( \mathcal{O}(mn^2) \) and space complexity \( \mathcal{O}(mn) \).

**Proof.** Note that the space requirement is identical, up to a constant factor, to the size of the instance, since utility matrix \( u \) is precisely of size \( m \times n \). The guarantee of \( m \)-\text{Greedy} being a direct consequence of Theorem 2, we focus on the complexity of the algorithm. The number of states in the dynamic program is clearly \( m \times n \) as we compute cost \( \mathcal{C}(i, k) \) for \( i = 1, \ldots, m \) and \( k = 1, \ldots, n \). Each state computation can be done in time \( \mathcal{O}(n) \), corresponding to the execution time of \( \text{Greedy} \) in one dimension, if items are sorted according to their densities. The sorting can be done in a preprocessing step of \( \mathcal{O}(mn \log n) \) before computing costs \( \mathcal{C}(i,..) \) and thus does not influence the overall complexity. To exhibit a feasible solution \( x \), we can backtrack the \( \mathcal{C}(i, k) \) from \( i = m \) to 1. More precisely, if we have kept in memory that the value of \( \mathcal{C}(i, k) \) is minimized by discarded item \( l \), we can, in time \( \mathcal{O}(n) \), add to current solution \( x \) all the fully-copied items till \( l \), and then inductively complete \( x \) according to \( \mathcal{C}(i-1, l) \). Thus, we can build a feasible solution \( x \) of cost at most the cost of \( m \)-\text{Greedy} in time \( \mathcal{O}(mn) \).
at most \((m + 1)\) distinct items.

4. Knapsack problems with easy constraints

We consider here some special structures of instances, and show the flexibility of algorithm \(m\)-Greedy to take advantage of them. In particular, assume that a subset \(\mathcal{M}\) of the knapsack constraints has a special structure for which an efficient algorithm \(\mathcal{A}\) is known, with a guarantee \(\alpha\). Without loss of generality we can reindex the constraints such that \(\mathcal{M}\) constitutes the first \(|\mathcal{M}|\) ones. We can then consider the variation of \(m\)-Greedy, called \(m\)-Greedy\((\mathcal{A})\), where algorithm \(\mathcal{A}\) is used to compute \(C(|\mathcal{M}|, k)\) for any item \(k\) as the recursive basis of \(m\)-Greedy. Theorem 2 of previous section implies that algorithm \(m\)-Greedy\((\mathcal{A})\) has a guarantee of \((\alpha + m - |\mathcal{M}|)\) for these instances.

In the remaining of this section we focus on the 2-dimensional knapsack problem, however the following discussion remains valid in higher dimensions. We have established in the previous section that \(m\)-Greedy is a 3-approximation for 2D-minKP, running in time \(O(n^2)\). As a consequence of the discussion above, we can state the following corollary, considering that the first constraint is "easy" to solve to optimality:

**Corollary 2.** If one knapsack relaxation admits an exact algorithm \(\mathcal{A}\) running in time \(t_{\mathcal{A}}(n)\), then \(m\)-Greedy\((\mathcal{A})\) is to a 2-approximation for 2D-minKP on this class of instances, running in time \(O(nt_{\mathcal{A}}(n))\).

In the literature, some special polynomial cases of the one dimensional Knapsack Problem have been identified. On these classes of instances, \(m\)-Greedy\((\cdot)\) leads to a 2-approximation. Among these classes we can cite

- **Divisible utilities.** Assuming that items are indexed in a non-decreasing order of their utility, an instance has divisible utilities if \(u_j\) divides \(u_{j+1}\) for \(j = 1, \ldots, n - 1\). This is the case for example if all utilities are powers of 2. Such instances of the bounded knapsack problem can be solved in time \(O(n^2\log n)\), see [18].

- **Cross ratio ordered.** An instance is cross ratio ordered if
\[
\frac{u_{j+1}}{u_j} \leq \left\lfloor \frac{f_{j+1}}{f_j} \right\rfloor \quad \forall j = 1, \ldots, n - 1.
\]
Such instances of the bounded knapsack problem can be solved in linear time, see [6].

- **Arithmetic weight sequences.** In this case the utility of any item \(j\) can be written as \(u_j = \alpha + j\beta\), with \(\alpha\) and \(\beta\) two integers. The unbounded knapsack problem can be solved in time \(O(n^8)\), see [5].

Knapsack problem with cardinality constraint is another example of particular 2D-minKP with an "easy" knapsack constraint. This problem was studied by [2] for the maximization version of the \(\{0, 1\}\) knapsack problem, the cardinality constraint imposing a limit on the number of selected objects. It hence corresponds to the special case of identical utilities in one dimension, that is \(u_{ij} = u_i\) for all item \(j\). For the minimization version, by analogy, we consider that cardinality constraint imposes to select at least \(k\) copies of items, i.e., \(\sum_j x_j \geq k\). The knapsack problem with an additional cardinal-
ity constraint remains \( \mathcal{NP}-\text{hard} \) ([2]). The relaxation with the cardinality constraint alone can be solved obviously by selecting the \( k \) cheapest objects. This optimal solution is found by Greedy, since the density of an item is simply its cost. It results that algorithm \( m\text{-Greedy} \) is a 2-approximation, running in linear time, for the bounded knapsack problem with an additional cardinality constraint.

Finally consider the Multiple Choice Knapsack Problem, where we are given a partition \( N_1 \cup \ldots \cup N_q = N \) of the items, and the additional constraints that at least one item is selected in each set \( N_k \).

More generally, we can consider a laminar family \( \mathcal{L} \) of subsets of \( N \) (\( \mathcal{L} \) is laminar if for all \( X,Y \in \mathcal{L} \), we have \( X \subseteq Y \) or \( Y \subseteq X \) or \( X \cap Y = \emptyset \)), each set \( L \in \mathcal{L} \) being associated with a requirement \( r_L \) and the additional constraint that

\[
\sum_{j \in L} x_j \geq r_L \quad \forall L \in \mathcal{L}.
\]

The laminar family \( \mathcal{L} \) can be represented by an in-tree, by convention rooted at set \( N \) ([17]). Without loss of generality, we can assume that the requirement of set \( L \) is greater than the sum of the requirements of its children, otherwise its constraint is redundant and can be ignored. By considering the requirement constraints in a topological order of the in-tree, from leaf sets to root \( N \), it is easy to see by induction that selecting greedily the cheapest items leads to an optimal solution. It follows that that algorithm \( m\text{-Greedy} \) is a 2-approximation for the knapsack problem with additional laminar cardinality constraints.

5. Filtering

Filtering is a quite common approximation technique. It consists in guessing, by exhaustive enumeration, a subset of elements belonging to an optimal solution, and then applying an approximation algorithm on the remaining, filtered, problem. Recently Pritchard [14] shows using filtering and disjunctive programming that for any \( \varepsilon > 0 \) there exists a polynomial-sized linear program for \( m\text{-minKP} \) with an integrality gap at most \((1 + \varepsilon)\). In this section we consider a very limited application of filtering, guessing only the most expensive item used in an optimal solution. Thus assume that we have guessed that at least one copy of item \( g \) belongs to a given optimal solution \( x^* \), while for any item \( j \) such that \( f_j > f_g \) we have \( x^*_j = 0 \). The residual knapsack problem \( IP_{\mid g} \) is the \( m\text{-minKP} \) instance restricted to items of cost at most \( f_g \), with residual demands \( D'_i = D_i - u_{ig} \) for \( i = 1, \ldots, m \) and with one less available copy of item \( g \). Denoting by \( \text{OPT}_{\mid g} \) the optimal value of the residual problem, it is readily verified that \( \text{OPT} \geq f_g + \text{OPT}_{\mid g} \).

Now consider that we use \( m\text{-Greedy} \) to solve the residual problem. The advantage of filtering is that any item of \( IP_{\mid g} \) costs at most \( f_g \). In the one dimensional case, the solution returned by Greedy costs thus at most \( \text{OPT}_{\mid g} + f_g \). More generally, using the result of Theorem 2, it is easy to see that \( m\text{-Greedy} \) returns a feasible solution for \( IP_{\mid g} \) of cost at most \( m\text{OPT}_{\mid g} + f_g \). Let \( \alpha = f_g / \text{OPT} \). Then the feasible solution returned for problem \( IP \) by the filtered \( m\text{-Greedy} \) algorithm, assuming that \( g \) is our guess, has a cost at most

\[
\mathcal{C}_F^G(g) \leq m\text{OPT}_{\mid g} + 2f_g
\]
\[ \leq (m - (m - 2)\alpha)OPT \]

The filtering technique considers each item \( j \) as a possible candidate for \( g \), applies \textit{m-Greedy} on the corresponding residual problem, and outputs the least cost \( \mathcal{C}_G^F(j) \) found in its course. Since \( \alpha > 0 \), filtering \textit{m-Greedy} with the most expensive item results in an approximation algorithm of guarantee \( m \). Observe that this improvement is achieved at the expense of a factor \( n \) in the time complexity of the algorithm, due to the outer loop guessing item \( g \). Also notice that this filtered algorithm performs substantially better when \( \alpha \) is close to 1, that is, when \( f_g \) is almost equal to the entire optimal cost. Indeed, the guarantee tends to 2 when \( \alpha \) tends to 1.

A natural idea is to combine the filtered \textit{m-Greedy} with another approximation algorithm performing well in the case where \( \alpha \) is close to 0, that is, when \( f_g \) represents a small fraction of OPT. To this end, we consider a classical LP-rounding algorithm, where the residual problem is solved using the linear relaxation of formulation \( IP_{\bar{g}} \) and the optimal fractional basic solution \( \bar{x} \) is rounded up. Since at most \( m \) variables can be fractional in a basic solution, it follows that this rounding provides a solution for the residual problem of cost at most \( \text{OPT}_{\bar{g}} + mf_g \). Then the feasible solution returned for problem \( IP \), assuming that \( g \) is our guess, costs at most

\[ \mathcal{C}_G^F(g) \leq \text{OPT}_{\bar{g}} + (m + 1)f_g \leq (1 + m\alpha)OPT. \]

Clearly for small values of \( \alpha \), the rounding algorithm performs quite well and tends to be optimal when \( \alpha \) tends to zero. Selecting the best solution returned by both filtered algorithms, we obtain by definition a cost at most \( \min \{ \mathcal{C}_G^F(g), \mathcal{C}_{LP}^F(g) \} \). This quantity is maximal when \( \alpha = 1/2 \), i.e., when a copy of \( g \) represents half the optimal cost, which leads to the following theorem:

\textbf{Theorem 3.} Combining the filtered \textit{m-Greedy} and LP-rounding algorithms leads to a \((1 + m/2)\)-approximation for \( mD\text{-minKP} \).

As previously noticed, the filtered \textit{m-Greedy} has a time complexity of \( O(mn^2) \). On the other hand, the time complexity of the filtered LP-rounding algorithm corresponds to solve to optimality \( n \) LP relaxations, using for instance Ellipsoid method. When \( m \) is fixed, the dual LP can be solved in linear time \( O(n) \) due to a result of Megiddo & Tami [13], the multiplicative coefficient growing exponentially with \( m \).

In the case of the 2D \(- \text{minKP} \) with an easy constraint considered in Section 4, for a guess \( g \), the greedy algorithm returns a solution for the residual problem of cost at most \( 2\text{OPT}_{\bar{g}} \). Thus in this case we have

\[ \mathcal{C}_G^F(g) = 2\text{OPT}_{\bar{g}} + f_g \leq (2 - \alpha)OPT. \]

Combining with the filtered LP-rounding algorithm, the worst case guarantee corresponds to \( \alpha = 1/3 \), resulting in a 5/3-approximation.

6. Conclusion

In this study, we proposed a greedy approximation algorithm for the minimization version of the multi-dimensional knapsack problem and investigated the possibilities to improve its guarantee. Al-
algorithm $m$-GREEDY has the particularity to consider the effective utilities of items in each dimension in its turn, and to select item by batch, i.e. several copies at a time, most items being fully copied or not selected at all. Observe that $m$-GREEDY considers an arbitrary sequence of relaxations to build its solution. One natural question is to determine if some particular sequences exist (and can be efficiently found) that lead to a better guarantee. For instance, if for some index $i$ we know that all problems $IP(i, k)$ have optimal value small with regards to OPT, say $\text{OPT}(i, k) \leq \frac{1}{2}\text{OPT}$, then the guarantee of $m$-GREEDY becomes $1/2i + (m-i) = m-i/2$.

One future research direction is to extend the primal-dual algorithm of Carnes and Shmoys [3] for the single dimension to multiple dimensions. This algorithm is based on a reformulation of the problem using flow cover inequalities. The same idea of using flow cover inequalities, when coupled with surrogate relaxations, lead to different reformulations for the multi-dimensional problem. We believe that it is interesting to study how these formulations can be solved with primal-dual algorithms.

References

Algorithm 1 Compute $\mathcal{C}(i,k)$ for $k \in N$ and $i \in \{1, \ldots, m\}$ for mD-minKP

Require: $\mathcal{C}(i-1,l)$ for all $l \in N$ and items indexed in non-increasing order of density $d_{il} = f_l / u_{il}$

$\mathcal{C}(i,k) \leftarrow \infty$ \{upper bound\}

if $u_{ik} \geq D_i$ then
  $\mathcal{C}(i,k) \leftarrow \mathcal{C}(i-1,k)$
else
  $b_k \leftarrow b_k - 1$
  $D_i \leftarrow D_i - u_{ik}$
  $x \leftarrow 0$ \{current greedy solution\}
  $\text{cost} \leftarrow f_k$ \{current greedy cost\}
  $\text{util} \leftarrow 0$ \{current greedy utility\}
  for $l = 1$ to $n$ do
    if $\text{util} + b_l u_{il} < D_i$ then
      \{item $l$ is fully-copied\}
      $x_l \leftarrow b_l$
      $\text{cost} \leftarrow \text{cost} + f_l b_l$
      $\text{util} \leftarrow \text{util} + u_{il} b_l$
    else
      \{item $l$ is discarded\}
      \{the upper bound $\mathcal{C}(i,k)$ is updated\}
      $a_l \leftarrow \left\lceil \frac{D_i - \text{util}}{u_{il}} \right\rceil - 1$
      $\mathcal{C}(i,k) \leftarrow \min \left\{ \begin{array}{l} \mathcal{C}(i,k) \\ \text{cost} + f_l a_l + \mathcal{C}(i-1,l) \end{array} \right\}$
    end if
  end for
end if
return $\mathcal{C}(i,k)$
