Approximate minimization algorithms for the 0/1 Knapsack and Subset-Sum Problem

Michael M. Gützer, Dieter Jungnickel *

Lehrstuhl für Diskrete Mathematik, Optimierung und Operations Research, Universität Augsburg, Universitätstrasse 14, D-86135 Augsburg, Germany

Received 1 January 1998; received in revised form 1 August 1999

Abstract

The well-studied 0/1 Knapsack and Subset-Sum Problem are maximization problems that have an equivalent minimization version. While exact algorithms for one of these two versions also yield an exact solution for the other version, this does not apply to \( \varepsilon \)-approximate algorithms. We present several \( \varepsilon \)-approximate Greedy Algorithms for the minimization version of the 0/1 Knapsack and the Subset-Sum Problem, that are also \( \varepsilon \)-approximate for the respective maximization version. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: 0/1 Knapsack Problem; Subset-Sum Problem; Approximation; Greedy Algorithms

1. Introduction

The objective of the NP-hard 0/1 Knapsack Problem (KP) is to fill a knapsack with upper capacity \( c \in \mathbb{N} \) with items having weight \( w_i \in \mathbb{N} \) and profit \( p_i \in \mathbb{N} \) \((i = 1, \ldots, n)\) such that the total weight of all inserted items does not exceed the capacity while their total profit is maximized.

KP

\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{n} p_i x_i \\
\text{s.t.} & \quad \sum_{i=1}^{n} w_i x_i \leq c \\
& \quad x_i \in \{0, 1\} \quad \text{for all } i = 1, \ldots, n.
\end{align*}
\]

* Corresponding author. Fax: +49-821-598-2200.
E-mail address: jungnickel@math.uni-augsburg.de (D. Jungnickel)
The decision variables \( x_1, \ldots, x_n \) indicate which items are inserted in the knapsack, i.e. \( x_i = 1 \Leftrightarrow \text{item} \ i \ \text{is inserted. Without loss of generality, we assume} \)

\[
w_i \leq c \quad \text{for all} \quad i = 1, \ldots, n
\]

and

\[
\sum_{i=1}^{n} w_i > c.
\]

Clearly, we can state the KP as the problem to minimize the profit of the items not inserted in the knapsack subject to the condition that their combined weight has to be at least \( d := \sum_{i=1}^{n} w_i - c \). We will refer to this definition as Minimization Knapsack Problem (MinKP).

MinKP

\[
\begin{align*}
\min \quad & \sum_{i=1}^{n} p_i y_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} w_i y_i \geq d, \\
& y_i \in \{0, 1\} \quad \text{for all} \quad i = 1, \ldots, n.
\end{align*}
\]

The MinKP variables \( y_1, \ldots, y_n \) indicate which items are inserted in the minimization knapsack, which means that they are not inserted in the maximization knapsack. As a consequence, to a solution \( x_1, \ldots, x_n \) of the KP we define the corresponding solution of the MinKP as \( y_i := 1 - x_i \quad (i = 1, \ldots, n) \), and vice versa. The NP-hard Subset-Sum Problem (SSP) and the respective Minimization SSP (MinSSP) denote the special case of the KP and MinKP where \( p_i = w_i \) for all \( i = 1, \ldots, n \).

2. Summary of approximation ratios

While exact algorithms for one of the presented maximization or minimization problems also yield an exact solution for the corresponding version, this is not always the case for \( \varepsilon \)-approximate algorithms. An \( \varepsilon \)-approximate algorithm is an algorithm that yields a solution with a relative deviation from the optimum of less than the approximation ratio \( \varepsilon \). This means that for every instance of the given maximization [minimization] problem with optimal objective value \( z^* \) the \( \varepsilon \)-approximate algorithm yields a solution value \( z \) which is greater equal than \( (1 - \varepsilon)z^* \) [less equal than \( (1 + \varepsilon)z^* \)].

Table 1 shows some known \( \varepsilon \)-approximate algorithms and compares them to the algorithms of this paper, which are typeset in italics. All the given ratios are tight.

We would like to mention that we have been informed about some unpublished improvements concerning the approximation of KP by A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger.

3. Approximate algorithms for the KP and MinKP

The algorithms Greedy and Critical Item as well as their approximation ratios are presented in [4, pp. 27–29]. Therefore, we only discuss the algorithm MinGreedy in this section. To simplify the discussion we assume that the items are sorted in ascending order according to their profit per weight, i.e.

\[
\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \cdots \leq \frac{p_n}{w_n}
\]

which can be done with a complexity of \( O(n \ln(n)) \).
Table 1
Summary of approximation ratios

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Approximation ratio for Maximization</th>
<th>Approximation ratio for Minimization</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>KP/MinKP</td>
<td>[1]; [4, p. 17]</td>
<td>$\frac{1}{2}$</td>
<td>$\infty$</td>
<td>O(n)</td>
</tr>
<tr>
<td>Critical item</td>
<td>[4, pp. 27–29]</td>
<td>$\frac{1}{2}$</td>
<td>$\infty$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>Greedy</td>
<td>[4, pp. 117–118]</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>MinGreedy</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>SSP/MinSSP</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\infty$</td>
<td>O(n)</td>
</tr>
<tr>
<td>First Fit/Heaviest Item</td>
<td>[4, pp. 118–119]</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>Greedy</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>MTGS</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>MinGreedy 1</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>MinGreedy 2</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
<tr>
<td>MinGreedy 3</td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>O(n ln(n))</td>
</tr>
</tbody>
</table>

*aImproved approximation ratio after a simple preprocessing with complexity O(n).*

We describe MinGreedy as a minimization algorithm. The main idea is almost identical to an algorithm proposed in [2] for the more general Capacitated Plant Allocation Problem (CPAP). In analogy to the Greedy Algorithm a solution is constructed by insertion of the items according to their profit per weight. The algorithm MinGreedy starts with a current solution by setting all decision variables to 0 and by marking all items as unprocessed. Then, the algorithm repeats two phases, the insertion and the completion phase, until all items are processed.

The insertion phase starts by inserting the unprocessed items one by one in the given order until the insertion of the heaviest unprocessed item would lead to a feasible solution. In the following completion phase, all items whose addition to the current solution would lead to a feasible solution are considered. Every new solution constructed this way is compared to the best solution so far and stored if it is better. Whenever an item is used for insertion or completion it is marked as processed.

Finally, to reach a complexity of O(n ln(n)) the algorithm has to store the best solution so far implicitly by storing its total weight and its completion item. To determine the final solution, the completion item is inserted in the knapsack and the other items are added one by one in the given order whenever the insertion does not exceed the final weight.

Algorithm 1 (MinGreedy).

1 [Initialization]
   $w := 0; \ p := 0;$ [Current solution]
2 $\tilde{w} := \sum_{i=1}^{\#} w_i; \ \tilde{p} := \sum_{i=1}^{\#} p_i;$ [Best solution so far]
3 $k := 1;$ [Completion item]
4 $S := \{1, \ldots, n\};$ [Set of unprocessed items]
5 while $S \neq \emptyset$ and $p < \tilde{p}$ do
6      while $\max\{w; i \in S\} < d - w$ do
7         $j := \min S; \ w := w + w_j; \ p := p + p_j; \ S := S \setminus \{j\}$ [Insertion phase]
8      od;
9      while $\max\{w; i \in S\} \geq d - w$ do
10     od; [Completion phase]
Find \( j \) with \( w_j = \max\{w_i; i \in S\}; \ S := S \setminus \{j\}; \)
if \( p + p_j < \hat{p} \) then \( w := w + w_j; \ \hat{p} := p + p_j; \ k := j \)
\[ \text{od} \]

\[ \text{[Determine final solution]} \]
\( y_k := 1; \ w := w_k; \)
for \( i := 1 \) to \( n \) do
  if \( i \neq k \) then
    if \( w + w_i \leq \hat{w} \) then \( y_i := 1; \ w := w + w_i \) else \( y_i := 0 \)
  \[ \text{fi} \]
\[ \text{od} \]

The overall complexity of \( O(n \ln(n)) \) is reached by using a sorting index \( i_j \ (j = 1, \ldots, n) \) based on the weight of the items with \( w_j \geq w_{j+1} \) for all \( j = 1, \ldots, n - 1 \). By the use of this index the computation of the maxima in the lines 7 and 10 becomes trivial and every item has to be considered exactly once in each phase. For practical purposes a postoptimization can be added at the end of \text{MinGreedy} to eliminate excess profit by going through the selected items in descending order and removing an item whenever feasible.

The following theorem establishes the approximation ratios of \text{MinGreedy} for the KP and MinKP. Labbé, Schmeichel and Hakimi already established the 1-approximation of their algorithm, that is only slightly different to \text{MinGreedy}, for the CPAP, which is a generalization of the MinKP (see [2]). Nevertheless, we present this result as part of the following theorem since we can give a much shorter proof for the special case of the MinKP.

**Theorem 1.** \text{MinGreedy} is 1-approximate for MinKP and \( \frac{1}{2} \)-approximate for KP.

**Proof** First, we prove the approximation ratio for MinKP. Let \( j \) be the item that completes the first feasible solution \( y_1, \ldots, y_n \) and \( p^* \) the optimal profit. Because of the sorting of the items we have

\[ \sum_{i=1}^{j-1} p_i y_i < p^*. \tag{10} \]

If \( p_j \leq p^* \) then the 1-approximation is obvious. Otherwise, item \( j \) can be eliminated from the MinKP instance without changing the optimal solution. As item \( j \) is used for completion, it is not inserted in the current solution. Hence, the final solution of \text{MinGreedy} has a profit not exceeding the profit of \text{MinGreedy} applied to the problem instance with items \( 1, \ldots, j - 1, j + 1, \ldots, n \) and capacity \( d \). This proves the MinKP approximation ratio by induction.

For the approximation of KP let \( j = \min\{i = 1, \ldots, n; \ \sum_{k=1}^{i} w_k \geq d\} \) denote the critical item, \( x_1, \ldots, x_n \) the solution of \text{MinGreedy} for KP and \( p^* \) the optimal profit for KP. Because of the sorting of the items it follows that

\[ \sum_{i=j}^{n} p_i > p^*. \tag{11} \]

Obviously, \text{MinGreedy} finds a solution for MinKP that is completed by item \( j \), which leads to a corresponding solution for KP with a profit of at least

\[ \sum_{i=1}^{n} p_i x_i \geq \sum_{i=j+1}^{n} p_i. \tag{12} \]
Because of condition (4) we have
\[ d \leq d + c - w_j = \sum_{i=1 \atop i \neq j}^{n} w_i. \tag{13} \]
Hence, there is at least one other feasible solution found by MinGreedy (either before or after the one completed by item \( j \)). In this solution item \( j \) is not inserted in the knapsack of the MinKP and therefore the corresponding KP knapsack contains at least item \( j \), which leads to
\[ \sum_{i=1}^{n} p_i x_i \geq p_j. \tag{14} \]
Finally, we have
\[ \sum_{i=1}^{n} p_i x_i \geq \max \left\{ \sum_{i=j+1}^{n} p_i, \sum_{i=j+1}^{n} p_i \right\} \geq \frac{1}{2} \left( \sum_{i=j+1}^{n} p_i + p_j \right) > \frac{1}{2} p^*, \tag{15} \]
which completes the proof.

It is easy to see that the approximation ratio of 1 for MinKP is tight. For the following series of problem instances with \( m \geq 3 \),
\[ d = m, \quad w_1 = p_1 = m - 2, \quad w_2 = m - 1, \quad p_2 = m, \quad w_3 = m - 1, \quad p_3 = m, \quad w_4 = 1, \quad p_4 = 2, \tag{16} \]
the optimal profit is clearly \( p^* = m + 2 \), and the MinGreedy solution yields a profit \( p = 2m - 2 \), with \( (p - p^*)/p^* \) arbitrarily close to 1 for \( m \to \infty \).

The ratio of \( 1/2 \) for KP is also exact, as we can see by the problem instances given in (16). The optimal profit \( p^* \) for KP with the capacity \( c = \sum_{i=1}^{n} w_i - d = 2m - 3 \) is \( p^* = 2m - 2 \). MinGreedy yields a profit \( p = m + 2 \). Hence, for \( m \to \infty \) we have \( (p^* - p)/p^* \to \frac{1}{2} \).

It is interesting to see that the trivial condition (4) is essential for the validity of Theorem 1. If we omit item 2 in (16) \( c \) is reduced to \( m - 2 \). This does not affect the optimal and the MinGreedy profit for MinKP. But the profit of the corresponding solution of MinGreedy for KP is reduced to a value of 2, clearly not \( 1/2 \)-approximate for the optimal KP profit of \( m - 2 \) for \( m > 6 \).

4. Approximate algorithms for the SSP and MinSSP

We start this section by some remarks on the known algorithms for the SSP and then proceed to the discussion of the three specializations of MinGreedy for the MinSSP. The algorithm First Fit/Heaviest Item for the SSP takes the items in the given order (not necessarily sorted) and inserts them one by one in the knapsack whenever this does not exceed the upper capacity. Finally, the total weight is compared to the weight of the heaviest item alone and the better one of the two solutions is returned. Greedy for the SSP works almost the same way as First Fit/Heaviest Item except that the items are sorted in decreasing order, i.e.
\[ w_1 \geq w_2 \geq \cdots \geq w_n. \tag{17} \]
Therefore, the final comparison with the heaviest item is obsolete. MTGS consists of \( n \) runs of Greedy using only the items \( \{i, \ldots, n\} \) in the \( i \)th run. The best solution of the \( n \) runs is returned. Again, we assume that the items are sorted as in (17).
The approximation ratio of First Fit/Heaviest Item for SSP is obvious, as this algorithm is identical to the Greedy Algorithm for KP, which also needs the final comparison with the heaviest item to reach the approximation ratio of $\frac{1}{2}$. This ratio is established in [1, pp. 27–29]. To see that First Fit/Heaviest Item is not $\varepsilon$-approximate as MinSSP algorithm for any $\varepsilon$, we consider the following problem series: $d = 1$, $w_1 = m$, $w_2 = 1$, $w_3 = m$. The optimal MinSSP weight is $w^* = 1$, but with $c = 2m$ First Fit/Heaviest Item returns a weight of $w = m$ for the MinSSP. Hence, we have $(w - w^*)/w^* \to \infty$ for $m \to \infty$.

The approximation ratios of Greedy and MTGS for SSP can be found in [1, pp. 117–119]. According to our knowledge these approximation ratios are currently the best publicly known for their respective complexity. Tsung-Chyan Lai proposed a heuristic with a complexity of $O(n \ln(n))$ for SSP in [3] and claimed that it was $\frac{5}{3}$-approximate. However, this is incorrect, which can be seen using the following example: $c = 18$, $w_1 = 10$, $w_2 = w_3 = 9$, $w_4 = 1$. Tsung-Chyan Lai’s algorithm B returns a weight of 11 that is clearly below two thirds of the optimal weight of 18.

Trivially, MTGS always returns at least the weight of Greedy. Therefore, to prove the validity and tightness of the MinSSP approximation ratio of 1 for both algorithms, it is sufficient to show that Greedy is at least 1-approximate and that MTGS is not more than 1-approximate.

**Theorem 2.** Greedy and MTGS are 1-approximate for MinSSP.

**Proof** We define $w_{n+1} := 0$ and $j := \min\{i = 1, \ldots, n + 1: w_i \leq d\}$ as the heaviest item with weight not more than $d$. If

$$\sum_{i=j}^{n} w_i \geq d$$

Greedy is 1-approximate, as the first element that cannot be inserted in the SSP knapsack has a weight of less than $d$. Otherwise, the optimal solution of MinSSP is trivially the smallest item with weight greater than $d$. But in this case Greedy returns the optimal solution, because all items except one with the optimal weight are inserted in the SSP knapsack. \qed

The tightness of the approximation ratio of 1 for both algorithms is shown by the following series of MinSSP instances: $d = m + 1$, $w_1 = 2m$, $w_2 = m + 1$, $w_3 = w_4 = m$. With $c = 4m$ MTGS yields a weight of $3m + 1$ for SSP and thus of $2m$ for MinSSP. The optimal MinSSP weight is $m + 1$ and hence the ratio of 1 is tight.

The approximation ratios of the two Greedy Algorithms for MinSSP may seem relatively high. In fact, by some minor modifications we can easily improve the approximation ratios of both algorithms.

**Theorem 3.** If $w_i \leq d$ for all $i = 1, \ldots, n$ MTGS is $\frac{1}{2}$-approximate for MinSSP.

**Proof** We define $w_{n+1} = 0$ and $j := \min\{i = 1, \ldots, n + 1: w_i \leq \frac{1}{2}d\}$ as the heaviest item with a weight of not more than $\frac{1}{2}d$. We can assume $j > 1$, otherwise all items have a weight of not more than $\frac{1}{2}d$ and the approximation is trivial.

If we have

$$w_1 + \sum_{i=j}^{n} w_i < d,$$

then every feasible MinSSP solution has to contain at least two of the items $1, \ldots, j - 1$. Because of $w_{j-1} > \frac{1}{2}d$ and the sorting of the items the optimal MinSSP weight is $w_{j-2} + w_{j-1}$. This is exactly the weight returned by the first run of Greedy.
Otherwise,
\[ w_1 + \sum_{i=j}^{n} w_i \geq d \tag{20} \]
implies that the items 2, \ldots, j - 1 are inserted in the SSP knapsack in the second run of Greedy. If \( w_1 = d \) the items \( j, \ldots, n \) are also inserted in the SSP knapsack and the optimal MinSSP weight \( d \) is returned, else one of the items \( j, \ldots, n \) cannot be inserted in the SSP knapsack. Because of \( w_i \leq \frac{1}{2} d \) for \( i = j, \ldots, n \) it follows
\[ c - \sum_{i=1}^{n} w_i x_i \leq \frac{1}{2} d \quad \Rightarrow \quad \sum_{i=1}^{n} w_i y_i \leq \frac{3}{2} d. \tag{21} \]

This completes the proof. \( \square \)

It is noteworthy that we do not need all \( m \) runs of Greedy in MTGS, but only the first two to reach the \( \frac{1}{2} \)-approximation of the MinSSP in Theorem 3. We will use this fact to improve Greedy without changing its complexity.

First, we have to ensure by a pre-processing that no item has a weight of more than \( d \). We can do that by finding \( j := \max \{ i = 0, 1, \ldots, n : w_i > d \} \) (\( w_0 := \infty \)) the lightest item with a weight of more than \( d \). We store the SSP solution (\( x_j = 0, x_i = 1 \) for all \( i \neq j \)) and start Greedy two times. In the first run we use the items \( j + 1, \ldots, n \) and \( c - \sum_{i=1}^{j} w_i \) as upper bound, in the second run only the items \( j + 2, \ldots, n \) and the same bound. In these problem instances conditions (4) and (5) might be violated, but this does neither affect the SSP nor the MinSSP approximation ratio of Greedy. Eventually, we take the best of the three solutions, i.e. the solutions from the pre-processing and from the two runs. The complexity added by this modification is \( O(n) \), which leads to a total complexity of \( O(n \ln(n)) \) and an approximation ratio of \( \frac{1}{2} \) for MinSSP.

By adding the same pre-processing routine as above we can construct a version of MTGS that is \( \frac{1}{2} \)-approximate for MinSSP. In both cases the approximation ratio of \( \frac{1}{2} \) is tight, which is shown by the problem instance \( c = 4m, w_1 = 2m, w_2 = m + 1, w_3 = w_4 = w_5 = m \). We have \( d = 2m + 1 \) and an optimal MinSSP weight of \( w^* = 2m + 1 \). Both algorithms yield an MinSSP weight of \( w = 3m \) and therefore \( (w - w^*)/w^* \to \frac{1}{2} \) for \( m \to \infty \).

The algorithm MinGreedy 1 which is the basis of the three greedy minimization algorithms for SSP is a specialization of MinGreedy for MinKP. The basic concept of an insertion and completion phase is used again, but the sorting is done according to (17) based on the weight. The main idea behind this approach is to insert the heaviest items first in the MinSSP knapsack in a similar way as the Greedy Algorithm inserts the heaviest items first in the SSP knapsack. Because there is no profit involved, we do not need to determine any maxima in the set of unprocessed items, but we can use the unprocessed item with the smallest index instead. Therefore, each of the two phases of insertion and completion is reduced from a while-loop to a single line.

**Algorithm 2 (MinGreedy 1).**

1. [Initialization]
2. \( w := 0; \) \quad [Current solution]
3. \( \bar{w} := \sum_{i=1}^{n} w_i; \) \quad [Best solution so far]
4. \( k := 1; \) \quad [Completion item]
5. for \( i := 1 \) to \( n \) do
6. \hspace{1em} if \( w + w_i < d \) then
7. \hspace{2em} \( w := w + w_i; \) \quad [Insertion]
The following theorem establishes the approximation ratios of MinGreedy 1 as minimization and as maximization algorithm.

**Theorem 4.** MinGreedy 1 is $\frac{1}{2}$-approximate for MinSSP and $\frac{1}{2}$-approximate for SSP.

**Proof** The $\frac{1}{2}$-approximation of SSP is a direct consequence of Theorem 1. For the MinSSP approximation we assume, without loss of generality, that

MinGreedy 1 returns a weight greater than $\frac{1}{2}d$. (22)

If the optimal solution contains only one item MinGreedy 1 is exact. Otherwise, we can assume that we have

$$w_i < d \quad \text{for all } i = 1, \ldots, n$$

(23)

because the items with a weight greater or equal $d$ are all processed as completion items at the beginning of the algorithm. We define $y_1, \ldots, y_n$ as the last solution that could be completed to a feasible solution in the algorithm, irrespective of the fact that it might not be stored due to a greater weight than the best weight so far. Furthermore, we set $j := \max \{i; y_i = 1\} \geq 2$. By construction of the algorithm and assumption (22) we have $w_j > \frac{1}{2}d$. Because $y_1, \ldots, y_n$ is the last completed solution it follows that

$$\sum_{i=1}^{j-1} w_i y_i + \sum_{i=j+1}^{n} w_i < d.$$  

(24)

Because of (23) item 1 is inserted in the current solution which implies $y_1 = 1$. Hence, we have $w_1 + \sum_{i=j+1}^{n} w_i < d$ by (24). In this case the optimal solution has to contain at least two items with a weight of at least $w_j$, which means that an optimal solution consists of item $j$ and $j - 1$. The MinGreedy 1 solution contains items 1 and $j$, whose total weight exceeds the optimal weight by less than $\frac{1}{2}d$. This proves the $\frac{1}{2}$-approximation for MinSSP. 

It is important to note that the $\frac{1}{2}$-approximation of MinSSP is even valid if condition (4) is violated, since we did not need this fact in the proof. The example $d = 2m + 2$, $c = 2m$, $w_1 = 2m$, $w_2 = w_3 = m + 1$ with optimal weight $2m + 2$ for MinSSP and $2m$ for SSP proves the tightness of the given ratios, as MinGreedy 1 returns a weight of $3m + 1$ for MinSSP and $m + 1$ for SSP.

In the same way as MTGS is based on Greedy, we can improve the approximation ratios of MinGreedy 1 by starting the algorithm several times, each time reducing the set of items used. MinGreedy 2 consists of two runs of MinGreedy 1. In the first run all items are used, in the second all items except the heaviest one, which means that the first item is directly inserted in the SSP knapsack. The better one of the two runs
is used as final solution. MinGreedy 3 is the equivalent to MTGS. In MinGreedy 3 there are \( n \) runs of MinGreedy 1. In the \( i \)th run only the items \( i, \ldots, n \) are used. Each run is started only, if the volume of all items used is at least \( d \).

The following simple lemma is quite useful for the proof of the given approximation ratios. It is based on the fact that, if there are at any point enough items of sufficiently small weight left, the MinGreedy 1 weight will exceed \( d \) by not more than the weight of the heaviest of these items.

**Lemma 1.** We consider the beginning of the for-loop (lines 5–11) and refer with \( i \) to the loop index and with \( w \) to the current weight. If

\[
w + \sum_{j=1}^{n} w_j \geq d,
\]

then the weight returned by MinGreedy 1 will exceed \( d \) by less than \( w_i \).

**Proof** If the current solution is completed by one of the items \( i, \ldots, n \) the assertion is obvious because of \( w_n \leq w_{n-1} \leq \cdots \leq w_i \). This has to happen in the course of the algorithms as the total weight of the items \( i, \ldots, n \) is greater equal the weight deficit \( d - w \) of the current solution.

**Theorem 5.** MinGreedy 2 is \( \frac{1}{2} \)-approximate for MinSSP and \( \frac{1}{3} \)-approximate for SSP.

**Proof** The approximation ratio for MinSSP is the same as that of MinGreedy 1, which was shown in Theorem 4. For the SSP, we can assume, without loss of generality, that

\[
\text{MinGreedy 2 returns a weight of less than } \frac{1}{2} c.
\]

We proceed to show that there exist at least two items with a weight of more than \( \frac{1}{4} c \). Because of Lemma 1 there is at least one such item. If this is the only such item we can apply Lemma 1 again after this (heaviest) item has been processed as first item. If item 1 is not inserted in the current solution the current weight is 0 and the remaining weight is that of the items 2, \ldots, \( n \). This leads to

\[
\sum_{i=2}^{n} w_i = \sum_{i=1}^{n} w_i - w_1 = d + c - w_1 \geq d.
\]

With \( w_2 \leq \frac{1}{4} c \) we can now apply Lemma 1 and get a contradiction to assumption (26). If item 1 is inserted in the current solution we can apply Lemma 1 in exactly the same way as above using (27).

Let \( j \geq 2 \) be defined as the lightest item with a weight of more than \( \frac{1}{4} c \), \( j = \max \{ i = 1, \ldots, n : w_i > \frac{1}{4} c \} \). It follows that

\[
\sum_{i=1}^{j-2} w_i < d.
\]

Otherwise, a solution would be completed by one of the items 1, \ldots, \( j-2 \) leading to an SSP solution containing the items \( j-1, \ldots, n \) with a weight of more than \( \frac{1}{2} c \).

Furthermore, it follows that

\[
\sum_{i=1}^{j-1} w_i \geq d,
\]

since otherwise item \( j-1 \) would be inserted in the current solution and after processing item \( j-1 \) we would have a current weight of \( \sum_{i=1}^{j-1} w_i \). With \( w_j \leq c \) from (4) and \( w \) as the current weight after processing item \( j \)
we get
\[ w + \sum_{i=j+1}^{n} w_i = \sum_{i=1}^{n} w_i \geq d \] (30)

exactly as in (27). With \( w_{j+1} \leq \frac{1}{3}c \) we can now apply Lemma 1 which leads again to a contradiction of assumption (26). By the same argument we can conclude that item \( j \) is not inserted into the current solution.

Finally, we consider the remaining volume \( \sum_{i=j+1}^{n} w_i \) after processing item \( j \). At that point we have a current volume of \( w = \sum_{i=1}^{j-1} w_i \). Because of \( w_{j+1} \leq \frac{1}{3}c \) and assumption (26) the sum of the current and remaining volume cannot exceed \( d \), otherwise we would get a contradiction to Lemma 1. Thus, we can conclude
\[ d > w \sum_{i=j+1}^{n} w_i = \sum_{i=1}^{j-2} w_i + \sum_{i=j+1}^{n} w_i \quad \text{implying that } w_{j-1} + w_j > c. \] (31)

As items \( j-1 \) and \( j \) are the lightest ones having a weight of more than \( \frac{1}{3}c \) there is no feasible solution for SSP that contains more than one of the items \( 1, \ldots, j \).

We now consider the SSP solution returned by the second run of MinGreedy 1. By construction this solution has to contain item 1. Either, we have
\[ w_1 + \sum_{i=j+1}^{n} w_i > c \] (32)

which leads to the situation of Lemma 1 after all the items \( 2, \ldots, j \) have been inserted in the MinSSP knapsack, since \( \sum_{i=2}^{j} w_i + \sum_{i=j+1}^{n} w_i \geq d \) and \( w_{j+1} \leq \frac{1}{3}c \), but this contradicts assumption (26). Otherwise, we have
\[ w_1 + \sum_{i=j+1}^{n} w_i \leq c, \] (33)
then all the items \( 2, \ldots, j \) are inserted into the MinSSP knapsack during the second run of MinGreedy 1 returning the optimal solution. \( \square \)

The example
\[ c = 3m - 1, \quad w_1 = 2m, \quad w_2 = 2m - 1, \quad w_3 = w_4 = w_5 = m \] (34)
with \( d = 4m \) has an optimal SSP weight of \( 3m - 1 \), whereas MinGreedy 2 returns a weight of \( 2m \). The approximation ratio of MinGreedy 2 for the MinSSP is not better than that of MinGreedy 1 as the following example proves:
\[ d = 2m, \quad w_1 = w_2 = 2m - 1, \quad w_3 = w_4 = m. \] (35)
The optimal weight \( w^* \) for this example is \( 2m \). MinGreedy 2 returns a weight of \( 3m - 1 \), arbitrarily close to \( (1 + \frac{1}{2})w^* \) for \( m \to \infty \).

Finally, we now establish the approximation ratios of MinGreedy 3.

**Theorem 6.** MinGreedy 3 is \( \frac{1}{4} \)-approximate for MinSSP and \( \frac{1}{3} \)-approximate for SSP.

**Proof** The approximation ratio for SSP is established by Theorem 5. Let \( y_1^*, \ldots, y_n^* \) be an optimal MinSSP solution with \( \{i_1, i_2, \ldots, i_l\} = \{i : y_i^* = 1\} \) such that \( i_1 < i_2 < \cdots < i_l \) and the optimal weight \( w^* = \sum_{i=1}^{l} w_i y_i^* \).
If \( l \leq 2 \) the \( i \)th run of \textit{MinGreedy 1} returns the optimal solution. Hence, we can assume \( l \geq 3 \) and focus now on the \( i \)th run of \textit{MinGreedy 1}. Because of \( l \geq 3 \) we know that item \( i_1 \) is inserted in the current solution as well as another item \( j_2 \) with \( j_2 \leq i_2 \).

\[
v := w_{i_1} + w_{j_2} + w_{i_2} \leq \frac{5}{4} w^*, \tag{36}
\]

we have either \( v \geq d \), then the current solution containing the items \( i_1 \) and \( j_2 \) is completed by item \( i_3 \) with a total weight of not more than \( \frac{5}{4} w^* \). Otherwise, we have \( v < d \). Then, \( w_{i_3} \leq w_{j_2} \) implies that \( l \geq 4 \) and there is a third item \( j_3 \) with \( j_3 \leq i_3 \) and thus with a weight of at least \( w_{i_3} \) inserted into the current solution. By \( w_{i_4} \leq \frac{1}{4} w^* \) and

\[
w_{i_1} + w_{j_2} + w_{i_3} + \sum_{k=i_4}^{n} w_k \geq w_{i_1} + w_{i_2} + w_{i_3} + \sum_{k=4}^{l} w_k = w^* \geq d
\]

we can apply Lemma 1 at the processing of item \( i_4 \), which proves the approximation. This leaves only the case

\[
v > \frac{5}{4} w^*. \tag{38}
\]

We can conclude by \( w_{i_3} \leq \frac{1}{2}(w^* - w_{i_1}) \) (because of the sorting) that

\[
\frac{5}{4} w^* < w_{i_1} + w_{j_2} + w_{i_3} \leq 2w_{i_1} + \frac{1}{2}(w^* - w_{i_1}) = \frac{1}{2} w^* + \frac{3}{2} w_{i_1}
\]

which leads to

\[
\frac{1}{2} w_{i_1} > \frac{3}{4} w^* \iff w_{i_1} > \frac{3}{4} w^*. \tag{40}
\]

Now we apply Theorem 4 to the remaining MinSSP (after the insertion of item \( i_1 \), using items \( i_1 + 1, \ldots, n \)). This remaining problem does not necessarily satisfy condition (4), but as we remarked earlier condition (4) is not needed to prove the \( \frac{1}{2} \)-approximation of \textit{MinGreedy 1} for MinSSP, anyway. Thus, we can conclude that the weight \( w \) returned by the \( i_1 \)th run of \textit{MinGreedy 1} is bounded by \( w_{i_1} \) plus \( \frac{3}{4} \) of the optimal weight \( w^* - w_{i_1} \) for the remaining problem

\[
w \leq w_{i_1} + \frac{3}{4} (w^* - w_{i_1}) = w^* + \frac{1}{2} (w^* - w_{i_1}) < \frac{5}{4} w^*. \tag{41}
\]

This completes the proof. \( \square \)

Example (34) shows that the two given approximation ratios of \textit{MinGreedy 3} are tight. For MinSSP and SSP \textit{MinGreedy 3} returns a weight of \( 5m - 1 \) and \( 2m \), respectively. As the optimal weight is \( 4m \) for MinSSP and \( 3m \) for SSP, the given ratios are exact. The number of \textit{MinGreedy 1} runs is also tight, in the sense that the approximation ratio of \( \frac{1}{2} \) cannot be reached by a constant number of runs. This is shown by the example \( \sum d = 2m, w_1 = \cdots = w_{n-2} = 2m - 1, w_{n-1} = w_n = m \). \textit{MinGreedy 3} returns the optimal solution with a weight of \( 2m \), but this weight is found not before the \((n-1)\)th run. The lowest weight returned by one of the runs \( 1, \ldots, n-2 \) is \( 5m - 1 \), which is obviously not \( \frac{1}{4} \)-approximate.

\textbf{Acknowledgements}

We would like to thank Andreas Enge for his valuable comments and suggestions on a previous version of this paper.
References