# Approximation Algorithms for Knapsack Problems with Cardinality Constraints[†]

Alberto Caprara[*] Hans Kellerer[**] Ulrich Pferschy[**]
David Pisinger[¶]

**Abstract**

We address a variant of the classical knapsack problem in which an upper bound is imposed on the number of items that can be selected. This problem arises in the solution of real-life cutting stock problems by column generation, and may be used to separate cover inequalities with small support within cutting plane approaches to integer linear programs. We focus our attention on approximation algorithms for the problem, describing a linear-storage Polynomial Time Approximation Scheme (PTAS) and a dynamic-programming based Fully Polynomial Time Approximation Scheme (FPTAS). The main ideas contained in our PTAS are used to derive PTAS for the knapsack problem and its multidimensional generalization which improve on the previously proposed PTAS. We finally illustrate better PTAS and FPTAS for the subset sum case of the problem in which profits and weights coincide.

## 1 Introduction

The classical *Knapsack Problem* (KP) is defined by a set $N := \{1, \ldots, n\}$ of items, each having a positive integer profit $p_j$ and a positive integer weight $w_j$, and by a positive integer knapsack capacity $c$. The problem calls for selecting the set of items with maximum overall profit among those whose overall weight does not exceed the knapsack capacity. KP has the following immediate *Integer Linear Programming* (ILP) formulation:

$$\text{maximize} \quad \sum_{j \in N} p_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j \in N} w_j x_j \leq c \tag{2}$$

$$x_j \in \{0, 1\}, \quad j \in N, \tag{3}$$

[†]Technical Report, DIKU, University of Copenhagen 98/4.

[*]DEIS, University of Bologna, viale Risorgimento 2, I-40136 Bologna, Italy, acaprara@deis.unibo.it
[**]Universität Graz, Institut für Statistik und Operations Research, Universitätsstr. 15, A-8010 Graz, Austria, {hans.kellerer,pferschy}@kfunigraz.ac.at
[¶]DIKU, University of Copenhagen, Univ.parken 1, DK-2100 Copenhagen, Denmark, pisinger@diku.dk

where each binary variable $x_j$, $j \in N$, is equal to 1 if and only if item $j$ is selected. The *Subset Sum Problem* (SSP) is the special case of KP arising when $p_j = w_j$ for each $j \in N$. For notational convenience, later in the paper we will use the notation $p(S) := \sum_{j \in S} p_j$ and $w(S) := \sum_{j \in S} w_j$ for some $S \subseteq N$.

KP has widely been studied in the literature, see the book of Martello and Toth [13] and the recent survey by Pisinger and Toth [19] for a comprehensive illustration of the problem. Among the several applications of KP, two very important ones are the following. First, KP is the subproblem which appears when instances of the one-dimensional Cutting Stock Problem are solved by column generation, see e.g. [22]. Second, the separation of cover inequalities in cutting-plane/branch-and-cut approaches to general ILPs calls for the solution of a KP, see e.g. [15].

The *m-Dimensional Knapsack Problem* ($m$-DKP) is the generalization of KP in which each item $j \in N$ has $m$ nonnegative integer weights $w_{1j}, \ldots, w_{mj}$ and $m$ positive integer knapsack capacities $c_1, \ldots, c_m$ are specified. The objective is to select a subset of the items with maximum overall profit among those whose weight does not exceed the knapsack capacity for any of the $m$ dimensions. Formally, the problem can be formulated as (1) subject to (3) and the capacity constraints

$$\sum_{j \in N} w_{ij} x_j \leq c_i, \quad i \in M, \tag{4}$$

where $M := \{1, \ldots, m\}$. The importance of this problem follows from its generality, namely it is a generic ILP in which all variables are binary and all coefficients nonnegative. In the sequel we refer to $m$-DKP by implicitly assuming that $m$ is *fixed*, i.e. not part of the problem input. This implies for instance that a time complexity of $O(n^m)$ will be considered polynomial.

In this paper we address a problem which is at the same time a generalization of KP and a special case of 2-DKP, namely the *k-item Knapsack Problem* ($k$KP), which is a KP in which an upper bound of $k$ is imposed on the number of items that can be selected in a solution. The problem can be formulated as (1)–(3) with the additional constraint

$$\sum_{j \in N} x_j \leq k, \tag{5}$$

with $1 \leq k \leq n$. The *Exact k-item Knapsack Problem* (E-$k$KP) is the variant of $k$KP where the number of items in a feasible solution must be exactly equal to $k$. Clearly E-$k$KP can be formulated as $k$KP by replacing (5) by

$$\sum_{j \in N} x_j = k. \tag{6}$$

Without loss of generality we assume $w_j \leq c$ for $j \in N$. Actually, for E-$k$KP we shall assume that, for each $j \in N$, $w_j$ plus the sum of the smallest $k-1$ weights of items in $N \setminus \{j\}$ does not exceed $c$. The *k-item Subset Sum Problem* ($k$SSP) and *Exact k-item Subset Sum Problem* (E-$k$SSP) are the special cases of $k$KP and E-$k$KP, respectively, in which item profits and weights coincide.

Observe that $k$KP and E-$k$KP can easily be transformed into each other. More precisely, any $k$KP instance can be solved as the following E-$k$KP. The value of $k$ is the same, whereas the profits and the weights of the original items are multiplied by $k$ and the new capacity is defined as $(k+1)c-1$, where $c$ is the original capacity. Finally, $k$ "dummy" items with profit and weight 1 are added. By definition, an item subset of cardinality, say, $k' \leq k$ is feasible for the original $k$KP instance if and only if it satisfies the knapsack constraint for the E-$k$KP instance, in which case it can be made feasible by adding $k-k'$ dummy items. The ranking by profit of the feasible solutions of the two instances is the same due to profit scaling. On the other hand, E-$k$KP can be solved as a $k$KP by adding a suitably large quantity $M$ to the item profits, e.g. $M := p(N)$. If the optimal value $z^*$ of $k$KP is smaller than $kM$ then E-$k$KP has no solution, otherwise its optimal solution value is $z^* - kM$. This latter construction cannot be applied to show that E-$k$SSP can be transformed into $k$SSP, in fact we are not aware of any simple transformation between the two problems.

$k$KP is the subproblem to be solved when instances of the Cutting Stock Problem with cardinality constraints are tackled by column generation techniques. For instance the problem appears where the number of pieces cut from each stock is bounded by a constant due to the limited number of knives. $k$KP also appears in processor scheduling problems on computers with $k$ processors and shared memory. The Collapsing Knapsack Problem presented by [20], which has applications in satellite communication where transmissions on the band require gaps between the portions of the band assigned to each user, can be solved to optimality by considering $n$ $k$KP problems defined on different capacities but with the same set of items.

Furthermore, $k$KP could replace KP in the separation of cover inequalities, as outlined in the following. In general, separation calls for a constraint in some class which is violated by the optimal solution of the current LP relaxation, see [15]. Therefore, separation is a recognition rather than an optimization problem. For cover inequalities, the exact solution of a KP allows for the determination of the *most* violated such inequality, if any exists, without any control on the number of variables with nonzero coefficient in the inequality, which is the same as the number of items selected by the optimal KP solution. On the other hand, within cutting-plane approaches it is typically better to separate inequalities with a small number of nonzero coefficients, mainly because such inequalities are easier to handle for the LP solvers. This would suggest to focus on the separation of violated cover inequalities having a small number of nonzero coefficients, which can be carried out by explicitly imposing a cardinality constraint on the KP to be solved. In fact, cover inequalities are typically "lifted" before being added to the current LP relaxation, increasing the number of nonzero coefficients, as well as the strength of the inequality. Nevertheless, a cover inequality with few nonzero coefficients before lifting will correspond to a set of items with large weights, which are more likely to be selected if an upper bound on the cardinality of the solution is imposed. If the weights of the items with nonzero coefficient in the inequality before lifting are large with respect to the other weights, it is known [15] that most of the lifting coefficients will be equal to 0, regardless of the lifting procedure used. Accordingly, a cover inequality with fewer nonzero coefficients before lifting will tend to have fewer nonzero coefficients also after.

It is well known that both KP and $m$-DKP are NP-hard but pseudopolynomially solvable through dynamic programming, therefore the same properties hold for $k$KP and E-$k$KP (actually, the fact that E-$k$KP is solvable in pseudopolynomial time follows from the structure of the dynamic programming recursion that solves $k$KP, which will be shown later). On the other hand, for *fixed* $k$, both $k$KP and E-$k$KP are polynomially solvable in $O(n^k)$ time by brute force. A natural question is whether these problems are *fixed-parameter tractable* [2], i.e. there is a constant $\alpha$ and an algorithm which solves either problem in time $O(n^\alpha f(k))$, where $f$ of course may have exponential growth. The results reported in [3] state however that the existence of such an algorithm is unlikely, since this would imply that the classes $FPT$ and $W[1]$ are equal (something similar to $P = NP$).

In this paper we will mainly address polynomial time approximation algorithms for $k$KP (and E-$k$KP). For a generic problem P we will denote by $z^*$ the optimal solution value and by $z^H$ the value of the solution returned by a heuristic algorithm $H$. Similarly, for a subinstance $S$ of P, $z^*_S$ and $z^H_S$ will denote the optimal and heuristic solution values, respectively, for the subinstance. Assume P is a maximization problem, and consider $\varepsilon \in (0,1)$. We say that $H$ is a $(1-\varepsilon)$ *approximation algorithm* if $z^H \geq (1-\varepsilon)z^*$ for all instances of P. A *Polynomial Time Approximation Scheme* (PTAS) for P is an algorithm taking as input an instance of P and a value $\varepsilon \in (0,1)$, delivering a solution of value $z^H \geq (1-\varepsilon)z^*$, and running in a time polynomial in the size of the P instance. If the running time is also polynomial in $1/\varepsilon$ the algorithm is called a *Fully Polynomial Time Approximation Scheme* (FPTAS).

A PTAS for KP was proposed by Sahni [21], and the first FPTAS by Ibarra and Kim [7], later on improved by Lawler [11], Magazine and Oguz [12] and Kellerer and Pferschy [8]. While PTAS for KP typically require only $O(n)$ storage, all the FPTAS are based on dynamic programming and their memory requirement increases rapidly with the accuracy $\varepsilon$, which makes them impractical even for relatively big values of $\varepsilon$. For SSP, the best PTAS requiring $O(n)$ storage is due to Fischetti [4]. FPTAS were developed by Lawler [11], Gens and Levner [6] and recently improved by Kellerer et al. [9]. Also the more general $m$-DKP (recall that we assume $m$ not to be part of the input) admits a PTAS, as shown by Oguz and Magazine [16] and Frieze and Clarke [5], whereas Korte and Schrader [10] proved that the problem does not admit any FPTAS unless $P = NP$. The latter results clearly imply that $k$KP has a PTAS, while leaving open the question about the existence of a FPTAS, as well as of a PTAS for E-$k$KP. Actually, the question of a FPTAS for $k$KP is even more interesting as Korte and Schrader [10] proved that even for the 2-dimensional KP such a scheme does not exist unless $P = NP$.

In this paper we adapt classical PTAS and FPTAS for KP and SSP to $k$KP, E-$k$KP and $k$SSP. As a byproduct of our analysis, we propose some modifications to the classical PTAS for KP and $m$-DKP. With these changes, one can achieve the same approximation guarantee with a considerably reduced running time.

## 2 Linear programming and a PTAS for kKP

We start by defining a simple approximation algorithm for $k$KP based on the LP relaxation of the problem, obtained by replacing constraints (3) by

$$0 \leq x_j \leq 1, \quad j \in N. \tag{7}$$

Denote the optimal solution value of this LP by $z^{LP}$.

**Lemma 1** *An optimal basic solution $(x^*)$ of the LP relaxation (1), (2), (5) and (7), has at most two fractional components. Let $J_1 := \{k : x_k^* = 1\}$. If the basic solution has two fractional components $x_i^*$ and $x_j^*$, supposing w.l.o.g. $w_j \leq w_i$, then $p(J_1) + p_i \geq z^*$ and the solution defined by $J_1 \cup \{j\}$ is feasible for kKP.*

**Proof.** If upper bounds on the variables are treated implicitly, there will be two basic variables $x_i^*$, $x_j^*$ in a basic LP solution, whereas the other variables will take an integer value. If $x_i^*$ and $x_j^*$ are both fractional, then $x_i^* + x_j^* = 1$, $p(J_1) + p_i x_i^* + p_j x_j^* = z^{LP}$ and $w_i x_i^* + w_j x_j^* = c - w(J_1)$. Therefore, if $w_j \leq w_i$, then $p_i \geq p_j$, otherwise one could improve the LP solution by setting $x_i = 0$ and $x_j = 1$. Hence, $p_i x_i^* + p_j x_j^* \leq p_i(x_i^* + x_j^*) = p_i$, showing that $z^* \leq z^{LP} \leq p(J_1) + p_i$. Moreover, $w_j = w_j(x_i^* + x_j^*) \leq w_i x_i^* + w_j x_j^* = c - w(J_1)$, which yields the feasibility of $J_1 \cup \{j\}$. □

Lemma 1 is immediately extended to E-$k$KP, for which the number of fractional components in the LP solution can be either 0 or 2.

The algorithm below is a rather straightforward generalization of the well-known 1/2 approximation algorithm for KP, see e.g. [13], and proceeds as follows.

**Algorithm $H^{\frac{1}{2}}$:**

1. Compute an optimal basic solution of the LP relaxation of $k$KP. Let $J_1$ and $J_F$ contain, respectively, the indices of the variables at 1 and of the fractional variables in the LP solution.

2. If $J_F = \emptyset$, return the (optimal) $k$KP solution $J_1$, of value $z^H := p(J_1)$.

3. If $J_F = \{i\}$ for some $i \in N$, return the best $k$KP solution among $J_1$ and $\{i\}$, of value $z^H := \max\{p(J_1), p_i\}$.

4. If $J_F = \{i, j\}$ for some $i, j \in N$ such that $w_j \leq w_i$, return the best $k$KP solution among $J_1 \cup \{j\}$ and $\{i\}$, of value $z^H := \max\{p(J_1) + p_j, p_i\}$.

**Proposition 2** *$H^{\frac{1}{2}}$ is a $1/2$ approximation algorithm for kKP and runs in $O(n)$ time.*

**Proof.** The time complexity is due to the fact that the LP relaxation of $k$KP can be solved in $O(n)$ time, as shown by Megiddo and Tamir [14]. (The method by Megiddo and Tamir is in fact more general and rather sophisticated, applying Lagrangian relaxation and a multidimensional search procedure.) The feasibility of the solution returned follows immediately from Lemma 1. As to the approximation ratio, if the algorithm terminates at Step 2., the solution returned is clearly optimal, otherwise, denoting by $z^{LP}$ the optimal LP solution value and letting $z^G := p(J_1)$ if the algorithm stops at Step 3., $z^G := p(J_1) + p_j$ if it stops at Step 4., one has $z^* \leq z^{LP} \leq z^G + p_i \leq 2z^H$. □

A class of instances for which the approximation can be arbitrarily close to 1/2 is defined by $n = 3$, $p_1 = 2$, $w_1 = 1$, $p_2 = p_3 = M$, $w_2 = w_3 = M$, $k = 3$ and $c = 2M$, where $M$ is some "big" integer. The optimal solution consists of items 2 and 3, whereas the best solution computed by $H^{\frac{1}{2}}$ selects item 1 and only one item from 2 and 3, which yields $z^H/z^* = \frac{M+2}{2M} \stackrel{M\to\infty}{\longrightarrow} \frac{1}{2}$.

The adaptation of $H^{\frac{1}{2}}$ to E-$k$KP is quite easy. Using the same notation as in the description of $H^{\frac{1}{2}}$, observe that in the E-$k$KP case $J_F$ has cardinality zero or two. In the second case, the heuristic solution returned is the best among $J_1 \cup \{j\}$ and $\{i\} \cup R_i$, where $R_i$ is the set of the $k-1$ items with smallest weight in $N \setminus \{i\}$, which can be determined in $O(n)$ time by partial sorting techniques [1].

Our PTAS for $k$KP (and E-$k$KP) is based on heuristic $H^{\frac{1}{2}}$, and has the same flavor as the classical approximation scheme by Sahni [21] for KP and by Oguz and Magazine [16], Frieze and Clarke [5] for $m$-DKP.

**Algorithm** $PTAS_{kKP}$:

1. Let $\varepsilon \leq 1/2$ be the required accuracy, and define $\ell := \min\{\lceil 1/\varepsilon \rceil - 2, k\}$. Initialize the solution $H$ to be the empty set and set the corresponding value $z^H$ to 0.

2. Consider each $L \subset N$ such that $|L| \leq \ell - 1$. If $w(L) \leq c$ and $p(L) > z^H$ let $H := L$, $z^H := w(L)$.

3. Consider each $L \subseteq N$ such that $|L| = \ell$. If $w(L) \leq c$, apply algorithm $H^{\frac{1}{2}}$ to the subinstance $S$ defined by item set $\{k \in N \setminus L : p_k \leq \min_{j\in L} p_j\}$, by capacity $c - w(L)$ and by the cardinality upper bound $k - \ell$. Let $T$ and $z^H_S$ denote the solution and the solution value returned by $H^{\frac{1}{2}}$. If $p(L) + z^H_S > z^H$ let $H := L \cup T$ and $z^H := p(L) + z^H_S$.

4. Return solution $H$ of value $z^H$.

The main difference to the scheme of Sahni [21] is the fact that in Step 3. we complete the partial solution $L$ by considering only items whose profit is *at most* equal to the smallest profit of the items in $L$, according to [5,16].

**Proposition 3** *$PTAS_{kKP}$ is a PTAS for kKP which runs in $O(n^{\lceil 1/\varepsilon \rceil - 1})$ time and requires linear space.*

**Proof.** As defined in Step 1., let $\ell := \min\{\lceil 1/\varepsilon \rceil - 2, k\}$. The time and space requirements follow from the fact that algorithm $H^{\frac{1}{2}}$ and the definition of subinstance $S$ by partial sorting require $O(n)$ time and are executed $O(n^\ell)$ times in Step 3., whereas in Step 2.

the algorithm considers $O(n^{\ell-1})$ subsets, for each one performing operations that clearly require $O(n)$ time.

What remains to be shown is that $PTAS_{kKP}$ is a $(1-\varepsilon)$ approximation algorithm. Clearly, if an optimal solution contains at most $\ell$ items, the solution returned by $PTAS_{kKP}$ is optimal. Otherwise, let $\{j_1^*, j_2^*, \ldots, j_\ell^*, \ldots\}$ be the set of items in an optimal solution ordered so that $p_{j_1^*} \geq p_{j_2^*} \geq \ldots \geq p_{j_\ell^*} \geq \ldots$. In one of the iterations of Step 3., the set $L$ considered by $PTAS_{kKP}$ is $L^* := \{j_1^*, j_2^*, \ldots, j_\ell^*\}$. Let $S^*$ be the corresponding subinstance, on which algorithm $H^{\frac{1}{2}}$ is applied, returning a solution of value $z^H_{S^*}$. The optimal solution value is clearly given by $z^* = p(L^*) + z^*_{S^*}$, where $z^*_{S^*}$ is the optimal solution value for $S^*$. As $z^H \geq p(L^*) + z^H_{S^*}$, it is sufficient to show that $p(L^*) + z^H_{S^*} \geq \frac{\ell+1}{\ell+2} z^*$, and then the claim follows from the definition of $\ell$. We distinguish between two cases.

(i) $p(L^*) \geq \frac{\ell}{\ell+2} z^*$: From Proposition 2 we get

$$\begin{aligned} p(L^*) + z^H_{S^*} &\geq p(L^*) + \frac{1}{2} z^*_{S^*} = p(L^*) + \frac{1}{2}\left(z^* - p(L^*)\right) \\ &= \frac{1}{2}\left(z^* + p(L^*)\right) \geq \frac{1}{2}\left(z^* + \frac{\ell}{\ell+2} z^*\right) = \frac{\ell+1}{\ell+2} z^*. \end{aligned}$$

(ii) $p(L^*) < \frac{\ell}{\ell+2} z^*$: Obviously, the smallest profit of an item in $L^*$, and hence all the profits of items in $S^*$, are smaller than $\frac{1}{\ell+2} z^*$. We can assume that the solution of the LP relaxation of subinstance $S^*$ is fractional, otherwise the solution returned by $PTAS_{kKP}$ is optimal. In particular, let $p_{i^*}$ be profit of the item of larger weight whose associated variable is fractional, and $p(J_1^*)$ be the sum of the profits of items with variable at 1 in the LP solution (see algorithm $H^{\frac{1}{2}}$). Due to Lemma 1 one has

$$\begin{aligned} z^* &= p(L^*) + z^*_{S^*} \leq p(L^*) + p(J_1^*) + p_{i^*} \\ &\leq p(L^*) + z^H_{S^*} + p_{i^*} \leq p(L^*) + z^H_{S^*} + \frac{1}{\ell+2} z^*, \end{aligned}$$

and hence $\frac{\ell+1}{\ell+2} z^* \leq p(L^*) + z^H_{S^*}$. □

A class of instances for which the approximation ratio can be arbitrarily close to $\frac{\ell+1}{\ell+2}$ for a given $\ell$ is defined by $n = \ell+3$, $p_1 = 2$, $w_1 = 1$, $p_2 = \ldots = p_n = M$, $w_2 = \ldots = w_n = M$, $k = n$, and $c = (\ell+2)M$, where $M$ is some "big" integer. The optimal solution consists of all $\ell+2$ big items, whereas the best solution computed by $PTAS_{kKP}$ selects item 1 and only $\ell+1$ big items, which yields $z^H/z^* = \frac{(\ell+1)M+2}{(\ell+2)M} \stackrel{M\to\infty}{\longrightarrow} \frac{\ell+1}{\ell+2}$.

Again, algorithm $PTAS_{kKP}$ can be easily modified to handle E-$k$KP. In particular, Step 2. can be skipped, whereas in Step 3. one has to replace the call to heuristic $H^{\frac{1}{2}}$ by a call of the corresponding heuristic for E-$k$KP. The call must in fact be preceded by a preprocessing of subinstance $S$, which removes the items $j$ such that $w_j$ plus the smallest $k-\ell-1$ weights of items in $S$ other than $j$ exceeds the capacity. Such a preprocessing can be done in $O(n)$ time by partial sorting techniques. The time and space complexity and approximation analysis of the algorithm obtained are essentially unchanged.

# 3 Improved PTAS for KP and $m$-DKP

The main difference between the PTAS proposed in the previous section and the PTAS presented in the literature for KP [21] and $m$-DKP [5,16] is the fact that we use procedure $H^{\frac{1}{2}}$ instead of simply rounding down the LP solution. This modification yields improved PTAS also for KP and $m$-DKP, in the sense that we get better approximations for a given asymptotic running time bound. In fact, for KP we can make additional considerations to further reduce the complexity of our scheme.

The straightforward adaptation of $PTAS_{kKP}$ to KP just requires running the algorithm with the $k$KP instance obtained from a KP by setting the cardinality upper bound $k := n$. This would already constitute an improvement with respect to the scheme proposed by Sahni [21]. As anticipated, we can do better, by exploiting the structure of optimal LP solutions of KP. Namely, it is well-known that an LP solution of KP can be determined by sorting the items by decreasing profit/weight ratios, and this ordering solves the LP relaxation of KP for any given capacity. Let $PTAS_{KP}$ be the algorithm derived from $PTAS_{kKP}$ as follows, assuming $\varepsilon \leq 1/3$ .

In Step 1., we set $\ell := \min\{\lceil 1/\varepsilon \rceil - 2, n\}$ ($\ell \geq 3$) and sort and store items according to increasing profits. Furthermore, items are also sorted both by decreasing weights and by decreasing profit/weight ratios, and their consecutive indices according to these sortings are stored in arrays $W$ and $PW$, respectively. This step requires $O(n \log n)$ time.

In Step 3., we replace $H^{\frac{1}{2}}$ by its (straightforward) counterpart for KP, that returns the best solution among the one defined by the set of items with variable at 1 in the solution of the LP relaxation and the one defined by the item with fractional variable (if any). Moreover, the enumeration of the subsets $L$ of cardinality $\ell$ and the corresponding computation of a heuristic solution is carried out in the following way. Let every subset $L$ consist of $\{i_1, i_2, \ldots, i_\ell\}$ and assume $p_1 \leq p_2 \leq \ldots \leq p_n$.

For $i_1 := 1, \ldots, n - \ell + 1$
    Let $S := \{1, \ldots, i_1 - 1\}$ and sort the items in $S$ in increasing order of profit/weight ratio in $O(n)$ time by using $PW$.
    For $i_{\ell-1} := i_1 + \ell - 2, \ldots, n - 1$
        For every $T \subseteq \{i_1 + 1, \ldots, i_{\ell-1} - 1\}$ with $|T| = \ell - 3$
            Let the array $R$ contain the items $\{i_{\ell-1} + 1, \ldots, n\}$ sorted in decreasing order of weight by using $W$.
            Let $i_\ell := R[1]$ and compute the LP solution and the associated heuristic solution for a knapsack with item set $S$ and capacity $c - W(L)$.
            For $i_\ell := R[2], \ldots, R[n - i_{\ell-1}]$
                Recompute the LP solution and the associated heuristic solution for a knapsack with item set $S$ and capacity $c - W(L)$ starting from the previous solution.

In an outer loop, we let the item $i_1$ with smallest profit in the current set $L$ go from 1 to $n - \ell + 1$. For each $i_1$, we let the item in $L$ with the second largest profit, i.e. item

$i_{\ell-1}$, go through all possible indices from $i_1+\ell-2$ to $n-1$, and iteratively consider all subsets $T$ of $(\ell-3)$ items within $\{i_1+1,\ldots,i_{\ell-1}-1\}$. These subsets, jointly with $i_1$ and $i_{\ell-1}$, form $O(n^{\ell-1})$ subsets to consider. For each such subset we let the largest item in $L$, $i_\ell$, go through all indices from $i_{\ell-1}+1$ to $n$, in *decreasing order of weight*. Now we can compute the LP solution, and therefore the heuristic one, consecutively for all values of $i_\ell$ by inserting the items in $S$ into a knapsack of capacity $c-w(L)$ according to their order. As soon as the capacity is filled, we compute the heuristic solution and choose the next $i_\ell$ which yields a capacity $c-w(L)$ not smaller than before. Therefore, we can continue the computation of the LP solution from the point it was stopped for the previous set $L$, and hence the heuristic solutions for *all* possible items $i_\ell$ can indeed be determined in linear time.

Overall, we get a running time bound of $O(n^\ell)$ for Step 3., whereas three calls to a sorting procedure are required in Step 1. By using the same analysis as for $PTAS_{kKP}$, we have

**Proposition 4** *$PTAS_{KP}$ is a PTAS for KP which runs in $O(n^{\lceil 1/\varepsilon\rceil-2}+n\log n)$ time and requires linear space.*

To compare our algorithm with the Sahni PTAS [21] notice that the running time complexity of the latter is $O(n^{\lceil 1/\varepsilon\rceil})$. In other words, for $\ell\geq 2$, to achieve an approximation ratio of $\frac{\ell+1}{\ell+2}$ the Sahni scheme requires a running time of $O(n^{\ell+2})$ whereas our algorithm runs in $O(n^\ell)$ time. E.g. for a still reasonable running time of $O(n^3)$ we get an approximation ratio of 4/5 compared to the previous 2/3. As both PTAS are practical only for very small values of $\ell$, this speedup by a factor of $O(n^2)$ is a considerable gain in performance and enables one to get a solution with an approximation ratio "improved by two levels" within the same asymptotic running time (cf. Table 1).

In order to extend our PTAS and the corresponding analysis to $m$-DKP for any fixed $m$, we initially modify procedure $H^{\frac{1}{2}}$ so as to get its obvious extension $H^{\frac{1}{m+1}}$, which solves the LP relaxation of $m$-DKP determining a basic solution with at most $m$ fractional components. Again, let $J_1$ and $J_F=\{i_1,\ldots,i_k\}$, $k\leq m$, contain respectively the indices of the variables at 1 and with fractional value in the solution. The solution returned by $H^{\frac{1}{m+1}}$ is the best one among $J_1,\{i_1\},\ldots,\{i_k\}$, of value $\max\{p(J_1),p_{i_1},\ldots,p_{i_k}\}$. It was shown by Megiddo and Tamir [14] that the LP relaxation of $m$-DKP can be solved in $O(n)$ time (recall that we assume $m$ to be fixed). Hence, an immediate counterpart of Proposition 2 is

**Proposition 5** *$H^{\frac{1}{m+1}}$ is a $1/(m+1)$ approximation algorithm for $m$-DKP and runs in $O(n)$ time.*

The corresponding PTAS, called $PTAS_{m-DKP}$, is obtained by modifying $PTAS_{kKP}$, defining $\ell:=\min\{\lceil m/\varepsilon\rceil-(m+1),n\}$ in Step 1., and replacing the call to $H^{\frac{1}{2}}$ by a call to $H^{\frac{1}{m+1}}$ in Step 3., after having removed the items whose weight exceeds the residual knapsack capacity on some of the dimensions.

**Proposition 6** *$PTAS_{m-DKP}$ is a PTAS for $m$-DKP which runs in $O(n^{\lceil m/\varepsilon\rceil-m})$ time.*

| KP: | Complexity | $O(n)$ | $O(n \log n)$ | $O(n^2)$ | $O(n^3)$ | ... | $O(n^\ell)$ |
|---|---|---|---|---|---|---|---|
| | Previous[21] | 1/2 | 1/2 | 1/2 | 2/3 | ... | $\frac{\ell-1}{\ell}$ |
| | Ours | 1/2 | 2/3 | 3/4 | 4/5 | ... | $\frac{\ell+1}{\ell+2}$ |

| $k$KP: | Complexity | $O(n)$ | $O(n^2)$ | $O(n^3)$ | ... | $O(n^\ell)$ |
|---|---|---|---|---|---|---|
| | Previous[5,16] | – | 1/3 | 1/2 | ... | $\frac{\ell-1}{\ell+1}$ |
| | Ours | 1/2 | 2/3 | 3/4 | ... | $\frac{\ell}{\ell+1}$ |

| 2-DKP: | Complexity | $O(n)$ | $O(n^2)$ | $O(n^3)$ | ... | $O(n^\ell)$ |
|---|---|---|---|---|---|---|
| | Previous[5,16] | – | 1/3 | 1/2 | ... | $\frac{\ell-1}{\ell+1}$ |
| | Ours | 1/3 | 1/2 | 3/5 | ... | $\frac{\ell}{\ell+2}$ |

Table 1: Comparison of our PTAS and by the previous ones for KP, $k$KP and 2-DKP.

**Proof.** Analogous to the proof of Proposition 3. In particular, in the analysis of the approximation ratio one has to show that $p(L^*) + z^H_{S^*} \geq \frac{\ell+1}{\ell+m+1} z^*$. This is done again by considering two cases, namely $p(L^*) \geq \frac{\ell}{\ell+m+1} z^*$ and $p(L^*) < \frac{\ell}{\ell+m+1} z^*$. $\square$
The comparison with the previously proposed PTAS for $m$-DKP [5,16] shows that our scheme guarantees the same approximation ratio with an asymptotic running time reduced by a factor of $n$.

We give an overview of our improvements in Table 1, where we report the approximation achieved for a fixed running time bound by our PTAS and by the previous ones for KP, $k$KP and 2-DKP. For KP we consider the Sahni PTAS for time complexities $\leq O(n^2)$ and the classical greedy algorithm for lower complexities.

# 4 Dynamic programming and an FPTAS for $k$KP

In this section we first describe how to solve $k$KP and E-$k$KP to optimality by a pseudopolynomial dynamic programming scheme which is an immediate adaptation of the schemes for KP. The presented scheme is then used to derive a FPTAS for both problems. Note that dynamic programming algorithms for KP with cardinality constraints are also considered in [17] and [18].

Let $b$ be an upper bound on the optimal solution value. A straightforward dynamic programming recursion which has time complexity $O(nkb)$ and space complexity $O(k^2 b)$, can be stated as follows. Denote by function $g_i(a, \ell)$ for $i = 1, \ldots, n$; $\ell = 1, \ldots, k$; $a = 0, \ldots, b$, the optimal solution value of the following problem:

$$g_i(a, \ell) := \min\left\{\textstyle\sum_{j=1}^{i} w_j x_j : \sum_{j=1}^{i} p_j x_j = a;\ \sum_{j=1}^{i} x_j = \ell;\ x_j \in \{0, 1\}, j = 1, \ldots, i\right\} \quad (8)$$

One initially sets $g_0(a, \ell) := +\infty$ for all $\ell = 0, \ldots, k$; $a = 0, \ldots, b$, and then $g_0(0, 0) := 0$. Then, for $i = 1, \ldots, n$ the entries of $g_i$ can be computed from those of $g_{i-1}$ by using the formula

$$g_i(a,\ell) := \min \begin{cases} g_{i-1}(a,\ell) \\ g_{i-1}(a-p_i,\ell-1)+w_i & \text{if } \ell>0;\ a\geq p_i \end{cases} \tag{9}$$

The optimal solution value of E-$k$KP is given by $\max_{a=0,\dots,b}\{a : g_n(a,k) \leq c\}$, whereas the optimal solution value of $k$KP is $\max_{a=0,\dots,b;\ell=0,\dots,k}\{a : g_n(a,\ell)\leq c\}$.

To achieve the stated time and space complexities we observe that only entries in $g_{i-1}(a,\ell)$ need to be stored in order to derive $g_i(a,\ell)$. However to determine the optimal solution vector we must save some previous entries as follows. Every modified entry $g_i(a,\ell)$ is generated from a previous entry $g_{i-1}(a-p_i,\ell-1)$ by adding a new item $i$ in recursion (9). Hence associated with $g_i(a,\ell)$ we store a pointer to the added item and a pointer to the previous entry in $g_{i-1}$. In this way, the representation of the solution vector can be seen as a directed rooted tree with nodes corresponding to items. Some nodes are linked to a function entry in $g_i$ while the nodes on the path from the root of a tree to such a linked node represent the corresponding set of chosen items. Clearly, each of the $kb$ function entries may contain a list of at most $k$ items which yields the $O(k^2b)$ space bound.

After each iteration of (9) we also have to release the nodes in $g_{i-1}$ which are not anymore part of a solution vector. Thus we keep a counter in each node indicating the number of immediate child nodes. Moving from stage $i-1$ to $i$ each counter associated with $g_{i-1}(a,\ell)$ is decreased by one. If the counter reaches 0, the node is put back in the pool of unused memory (organized as a garbage list) and its predecessor's counter is also decreased with the obvious recursive continuation.

To get the time complexity of $O(nkb)$ observe that the recursion (9) demands $nkb$ operations, and at each update only one counter is increased in some node. Hence, totally we can also have at most $nkb$ decrements of counters (and thus freeings of nodes) throughout the algorithm.

By using the dynamic programming scheme above, we can devise a simple FPTAS for KP and $k$KP, referred to as $FPTAS_{kKP}$. Let $z^H$ be the solution value returned by heuristic $H^{\frac{1}{2}}$. Scale the item profits space by replacing each value $p_j$ by $q_j := \left\lceil \frac{p_j k}{z^H \varepsilon} \right\rceil$, where $\varepsilon$ is the accuracy required, and set the upper bound $b$ for the new instance to $2\lceil k/\varepsilon\rceil + k$ (this upper bound is correct since $2z^H$ is an upper bound for the original instance and the optimal solution values before and after round up differ by at most $k$). Then apply the above dynamic programming scheme, and return as heuristic solution the optimal solution for the scaled profits. It is easy to see that

**Proposition 7** *$FPTAS_{kKP}$ is an FPTAS for KP which runs in $O(nk^2/\varepsilon)$ time and requires $O(n+k^3/\varepsilon)$ space.*

**Proof.** The time and space requirements are easily computed by plugging in the value of $b$ in the time and space requirements of the dynamic programming scheme. We still have to show that the solution value returned is within $(1-\varepsilon)$ of the optimum. Let $T^*$ and $\tilde{T}^*$ be an optimal item set for the original and for the scaled problem, respectively, and observe that $\tilde{T}^*$ is a feasible solution of the original problem. Using the notation $q(S) := \sum_{j \in S} q_j$ we further observe that $q(\tilde{T}^*) \geq q(T^*)$. Using the fact that

$$\frac{z^H \varepsilon}{k}(q_j - 1) < p_j \leq \frac{z^H \varepsilon}{k} q_j$$

we obtain that

$$p(T^*) - p(\tilde{T}^*) \leq \frac{z^H \varepsilon}{k} \left( q(T^*) + |\tilde{T}^*| - q(\tilde{T}^*) \right) \leq \frac{z^H \varepsilon}{k} |\tilde{T}^*| \leq \varepsilon z^*. \qquad \square$$

It is straightforward to check that the above scheme works for E-$k$KP as well.

Note that the scheme above may further be improved by using more complicated techniques as given in [11]. However, it is beyond the scope of this paper to elaborate all the details presented in that paper.

# 5 Approximation algorithms for $k$SSP

As $k$SSP is a special case of $k$KP, all the approximation algorithms presented in the previous section are suited for $k$SSP as well. In fact, by exploiting the fact that weights and profits coincide, it is possible to derive considerably improved approximation schemes for this latter problem. The key result for the derivation of such algorithms is Lemma 8 below.

All the algorithms presented in this section split the set of items into two subsets. More precisely, let $\varepsilon \in (0, 1)$ be the accuracy required: the set of *small* items $S$ contains the items whose weight does not exceed $\varepsilon c$, whereas the set of *large* items $L$ contains the remaining ones, whose weight is larger than $\varepsilon c$. The first obvious observation is that *at most* $\ell := \min\{\lceil 1/\varepsilon \rceil - 1, k\}$ large items can be selected by a feasible solution. On the other hand, an accuracy of $\varepsilon$ in the solution of the subinstance defined by the items in $L$ is sufficient to guarantee the same accuracy for the overall instance, as explained in the sequel.

We start by describing a simple linear-time greedy procedure $SUB(c', k')$ which returns a feasible solution of cardinality at most $k'$ for the subinstance defined by the small items and capacity $c'$:

**Procedure** $SUB(c', k')$**:**

1. Let $T$ be the set of the $k'$ items in $S$ with largest weight.

2. If $w(T) \leq c'$, return $T$.

3. Otherwise, remove iteratively an arbitrary item from $T$ until $w(T) \leq c'$ and then return $T$.

A *large item heuristic* for $k$SSP is a heuristic algorithm structured as follows. For each value $\ell' \in \{0, \ldots, \ell\}$ the algorithm determines a heuristic solution as $L(\ell') \cup S(\ell')$, where $L(\ell')$ is a solution of cardinality $\ell'$ for the subinstance defined by the large items, while $S(\ell')$ is the solution returned by $SUB(c - w(L(\ell')), k - \ell')$. Then the algorithm outputs the best solution among the $\ell + 1$ computed, of value, say, $z^H$. For $\ell' = 0, \ldots, \ell$, let $z^H(\ell') := w(L(\ell'))$ and denote by $z^*(\ell')$ the value of the best solution of cardinality $\ell'$ for the $k$SSP subinstance defined by the large items.

**Lemma 8** *Let $H$ be a large item heuristic for $k$SSP. If $z^H(\ell') \geq (1 - \varepsilon) z^*(\ell')$ for all $\ell' \in \{0, \ldots, \ell\}$, then $H$ is a $(1 - \varepsilon)$ approximation algorithm.*

**Proof.** Let $\ell^*$ be the number of large items in an optimal solution of $k$SSP. Let $z_L^*$ denote the total weight of these $\ell^*$ large items and $z_S^*$ the weight of the corresponding small items, respectively. Of course, $z^* = z_L^* + z_S^*$ and $z_L^* \leq z^*(\ell^*)$.

Consider the iteration of $H$ for which $\ell' = \ell^*$. If procedure $SUB(c - w(L(\ell^*)), k - \ell^*)$ performs Step 3., i.e. some small items must be removed in order to satisfy the capacity constraint, then

$$z^H \geq z^H(\ell^*) + w(S(\ell^*)) \geq (1 - \varepsilon)c \geq (1 - \varepsilon) z^*.$$

Otherwise, procedure $SUB(c - w(L(\ell^*)), k - \ell^*)$ stops at Step 2., i.e. the $k - \ell^*$ largest small items do not completely fill the capacity. In this case, $z_S^* \leq w(S(\ell^*))$, and hence

$$z^H \geq z^H(\ell^*) + w(S(\ell^*)) \geq (1 - \varepsilon) z^*(\ell^*) + z_S^* \geq (1 - \varepsilon) z_L^* + z_S^* \geq (1 - \varepsilon) z^*. \qquad \square$$

As an illustration of the above result, we present a simple 3/4 approximation linear-time heuristic $H^{\frac{3}{4}}$.

**Algorithm $H^{\frac{3}{4}}$:**

1. Initialize $L := \{j \in N : w_j > c/4\}$, $S := \{j \in N : w_j \leq c/4\}$, and $\ell := \min\{3, k, |L|\}$.

2. Let $H(0)$ be the solution returned by $SUB(c, k)$.

3. Let $H(1)$ be the solution defined by the union of the largest item $i$ in $L$ and the solution returned by $SUB(c - w_i, k - 1)$.

4. If $\ell < 2$ go to 6. Otherwise, let $i, j$ be the two largest items in $L$ with weight $\leq c/2$. If two such items exist let $L(2) := \{i, j\}$, otherwise let $L(2) := \emptyset$.
   Let $h$ be the smallest item in $L$ and $k$ be the smallest item in $L$ with weight $\geq c/2$. If such an item $k$ exists and $w(L(2)) < w_h + w_k \leq c$, update $L(2) := \{h, k\}$.
   If $L(2) \neq \emptyset$, let $H(2)$ be the solution defined by the union of $L(2)$ and the solution returned by $SUB(c - w(L(2)), k - 2)$, otherwise let $H(2) := \emptyset$.

5. If $\ell < 3$ go to 6. Otherwise, let $L(3)$ be the set of the three smallest items in $L$. If $w(L(3)) \leq c$, let $H(3)$ be the solution defined by the union of $L(3)$ and the solution returned by $SUB(c - w(L(3)), k - 3)$, otherwise let $H(3) := \emptyset$.

6. Return the best solution among $H(\ell')$, $\ell' = 0, 1, 2, 3$.

**Proposition 9** *$H^{\frac{3}{4}}$ is a $3/4$ approximation algorithm for kSSP and runs in $O(n)$ time.*

**Proof.** The running time bound follows from the fact that SUB can be performed in linear time by standard partial-sorting algorithms. To complete the proof, one can readily check that for every $\ell' = 0, 1, 2, 3$ the large items selected have an overall weight at least equal to $3z^*(\ell')/4$. Then the claim follows from Lemma 8. $\square$

The best known PTAS for SSP requiring only linear storage is due to Fischetti [4], and can be briefly described as follows. For a given accuracy $\varepsilon$, the set of items is subdivided into small and large items as defined above. Furthermore, the set $L$ of large items is partitioned into a *minimal* number of $q$ buckets $B_1, \ldots, B_q$ such that the difference between the smallest and the biggest item in a bucket is at most $\varepsilon c$. Clearly, $q \leq 1/\varepsilon - 1$. A $q$-tuple $(\nu_1, \ldots, \nu_q)$ is called *proper* if there exists a set $\tilde{L} \subseteq L$ with $|\tilde{L} \cap B_i| = \nu_i$ $(i = 1, \ldots, q)$ and $w(\tilde{L}) \leq c$. Then a procedure LOCAL returns for any proper $q$-tuple a subset $\tilde{L} \subseteq L$ with $|\tilde{L} \cap B_i| = \nu_i$ $(i = 1, \ldots, q)$ and $w(\tilde{L}) \leq c$ such that $w(\tilde{L}) \geq (1-\varepsilon)c$ or $w(\tilde{L}) \geq w(\bar{L})$ for each $\bar{L} \subseteq L$ with $|\bar{L} \cap B_i| = \nu_i$ $(i = 1, \ldots, q)$ and $w(\bar{L}) \leq c$. Procedure LOCAL is completed by adding small items in a greedy way. Since the $q$-tuple which corresponds to an optimal solution is not generally known, all possible values for $\nu_1, \ldots, \nu_q$ are tried. Therefore, only the current best solution value produced by LOCAL has to be stored. Any call of LOCAL can be done in linear time and an upper bound for the number of proper $q$-tuples is given by $2\lceil\sqrt{1/\varepsilon}\rceil - 3$ which gives a running time bound of $O(n^{2\lceil\sqrt{1/\varepsilon}\rceil-2})$, whereas the memory requirement is only $O(n)$, see [4]. In fact, a more careful analysis shows that the time complexity is also bounded by $O(n \log n + \phi(1/\varepsilon))$, where $\phi$ is a suitably-defined (exponentially growing) function. Moreover, a (simplified) variant of the scheme runs in $O(n + (1/\varepsilon)\log^2 1/\varepsilon + \phi(1/\varepsilon))$, but requires $O(n + 1/\varepsilon)$ space.

It is easy to adapt the above scheme and its variant to $k$SSP. Since the number of large items in a set $\tilde{L}$ which corresponds to $(\nu_1, \ldots, \nu_q)$ is just $\nu := \sum_{i=1}^{q} \nu_i$, we only have to discard the tuples such that $\nu > k$ and to replace the call to a greedy algorithm in LOCAL by a call to $SUB(c - w(\tilde{L}), k - \nu)$. Time and storage requirements remain the same. Let $PTAS_{kSSP}$ be the resulting scheme and $PTAS'_{kSSP}$ its variant. The following propositions follow immediately from the results of [4] and Lemma 8.

**Proposition 10** *$PTAS_{kSSP}$ is a PTAS for kSSP which runs in $O(\min\{n^{2\lceil\sqrt{1/\varepsilon}\rceil-2}, n \log n + \phi(1/\varepsilon)\})$ time and requires linear space.*

**Proposition 11** *$PTAS'_{kSSP}$ is a PTAS for kSSP which runs in $O(n + (1/\varepsilon)\log^2 1/\varepsilon + \phi(1/\varepsilon))$ time and requires $O(n + 1/\varepsilon)$ space.*

We conclude this section by showing an exact algorithm and a FPTAS for $k$SSP. At first a special dynamic programming algorithm is presented.

Let $g_i(d)$ for $i = 1, \ldots, n$; $d = 0, \ldots, c$ be the optimal solution value of the following $k$SSP defined on the first $i$ items and by knapsack capacity $d$:

$$g_i(d) := \min\left\{\ell : \textstyle\sum_{j=1}^{i} w_j x_j = d;\ \sum_{j=1}^{i} x_j = \ell;\ x_j \in \{0,1\}, j = 1, \ldots, i\right\} \tag{10}$$

where $g_i(d) := n+1$ if no solution satisfies the two constraints. One initially sets $g_0(0) := 0$, and $g_0(d) := n + 1$ for $d = 1, \ldots, c$. Then, for $i = 1, \ldots, n$ the entries of $g_i$ are computed from those of $g_{i-1}$ by using the recursion

$$g_i(d) := \min \begin{cases} g_{i-1}(d) \\ g_{i-1}(d - w_i) + 1 & \text{if } d > w_i \end{cases} \tag{11}$$

and the optimal solution value of $k$SSP is given by $z^* = \max_{d=0,\ldots,c}\{d : g_n(d) \leq k\}$. This means that we compute for every reachable subset value the minimum number of items summing up to this value. The recursion has time complexity $O(nc)$ and space complexity $O(kc)$, improving by a factor of $k$ over the recursion for $k$KP. Indeed, we solve the problem in $O(nc)$ for all cardinalities $k' \leq k$ and all capacities $c' \leq c$.

Unfortunately, we could not find a way to construct an FPTAS directly from this dynamic programming scheme. However, most classical FPTASs can be extended to the $k$SSP by "expanding by a factor of $k$". In the following we will briefly explain such an approach, called $FPTAS_{kSSP}$.

Given the accuracy $\varepsilon$, we partition the item set into big and small items as above, consider the subinstance defined by the big items in $L$, and perform the dynamic programming scheme (9) on intervals instead of all possible weight sums. Therefore, we let $\ell := \min\{\lceil 1/\varepsilon \rceil - 1, k\}$ and partition the range of values $[0, c]$ into $\lceil 1/\varepsilon \rceil$ intervals of equal length $c\varepsilon$. Instead of determining every reachable weight sum in $[0, c]$ we keep at most $2\,\ell$ values for each interval. In particular we consider a new item $i$ by adding $w_i$ to all currently stored weight sums, but then keep only the *smallest* and the *largest* weight sums for every interval and for every cardinality $\ell' = 1, \ldots, \ell$ of the associated subset. The corresponding modification of the recursion (9) is easy to implement.

After performing the dynamic programming algorithm on the large items, consider for each $\ell' = 0, \ldots, \ell$ the item set $L(\ell')$ corresponding to a value in a dynamic programming interval with largest weight among all such sets of cardinality $\ell'$, and complete this partial solution by adding the items returned by $SUB(c - w(L(\ell')), k - \ell')$. The final solution is the best among these.

It is shown by induction in [9] that this yields an FPTAS. In fact, one can show that after considering a new item, for every weight sum in an optimal dynamic programming scheme there exist an upper and a lower bound in the above dynamic programming array which differ by at most $c\varepsilon$.

Obviously, the size of the above dynamic programming array is $O(k/\varepsilon)$ and each entry contains a subset of at most $\ell$ items. This yields the following improvement on Proposition 7.

**Proposition 12** *$FPTAS_{kSSP}$ is an FPTAS for kSSP which runs in $O(n\,\ell/\varepsilon)$ time and requires $O(n+\ell^2/\varepsilon)$ space.* □

Assuming the more natural case of $k \le 1/\varepsilon$, one has an improvement by a factor of $k$ in both time and space complexity. Again, this scheme may further be improved by using more sophisticated techniques e.g. from [9], but presenting all the details is beyond the scope of this paper.

**Acknowledgements:**
The hospitality of the Universities of Bologna and Graz are gratefully acknowledged. The work of the first author was partially supported by CNR and MURST, Italy. The work of the fourth author was partially supported by the EC Network DIMANET through the European Research Fellowship No. ERBCHRXCT-94 0429.

# References

[1] N. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, "Time Bounds for Selection", *Journal of Comput. Syst. Sci.* **7** (1973), 448–461.

[2] R.G. Downey, M.R. Fellows, "Fixed-Parameter Tractability and Completeness I: Basic Results", *SIAM Journal on Computing* **24** (1995), 203–225.

[3] R.G. Downey, M.R. Fellows, "Fixed-Parameter Tractability and Completeness II: On Completeness for $W[1]$", *Theoretical Computer Science* **141** (1995), 109–131.

[4] M. Fischetti, "A New Linear Storage, Polynomial-Time Approximation Scheme for the Subset-Sum Problem", *Discrete Applied Mathematics* **26** (1990), 61–77.

[5] A.M. Frieze, M.R.B. Clarke, "Approximation Algorithms for the $m$-Dimensional 0-1 Knapsack Problem:Worst-Case and Probabilistic Analyses", *European Journal of Operational Research* **15** (1984), 100-109.

[6] G.V. Gens, E.V. Levner, "Fast Approximation Algorithms for Knapsack Type Problems", in K. Iracki, K. Malinowski, S. Walukiewicz (eds.) *Optimization Techniques, Part 2*, Lecture Notes in Control and Information Sciences 23 (1980), Springer, Berlin, 185–194.

[7] O.H. Ibarra, C.E. Kim, "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems", *Journal of the ACM* **22** (1975), 463–468.

[8] H. Kellerer, U. Pferschy, "A New Fully Polynomial Approximation Scheme for the Knapsack Problem", *Technical Report*, Faculty of Economics, University Graz, submitted.

[9] H. Kellerer, R. Mansini, U. Pferschy, M.G. Speranza, "An efficient fully polynomial approximation scheme for the subset-sum problem", *Technical Report*, Faculty of Economics, University Graz, submitted, see also *Proceedings of the 8th ISAAC Symposium, Springer Lecture Notes in Computer Science* **1350**, 394–403, 1997.

[10] B. Korte, R. Schrader, "On the Existence of Fast Approximation Schemes", *Nonlinear Programming* **4** O.L. Mangasarian, R.R. Meyer, S.M. Robinson (ed.), Academic Press 1981, 415–437.

[11] E.G. Lawler, "Fast Approximation Algorithms for Knapsack Problems", *Mathematics of Operations Research* **4** (1979), 339–356.

[12] M.J. Magazine, O. Oguz, "A Fully Polynomial Approximation Algorithm for the 0-1 Knapsack Problem", *European Journal of Operational Research* **8** (1981), 270–273.

[13] S. Martello, P. Toth, *Knapsack Problems*, J. Wiley & Sons, Chichester, 1990.

[14] N. Megiddo, A. Tamir, "Linear Time Algorithms for some Separable Quadratic Programming Problems", *Operations Research Letters* **13** (1993), 203–211.

[15] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, J. Wiley & Sons, New York, 1988.

[16] O. Oguz, M.J. Magazine, "A Polynomial Time Approximation Algorithm for the Multidimensional 0-1 Knapsack Problem", Working Paper, University of Waterloo, 1980.

[17] U. Pferschy, D. Pisinger, G.J. Woeginger, "Simple but Efficient Approaches for the Collapsing Knapsack Problem", *Discrete Applied Mathematics* **77** (1997), 271–280.

[18] D. Pisinger, "Strongly Correlated Knapsack Problems are Trivial to Solve", *Proceedings CO96*, Imperial College of Science, Technology and Medicine, London 27–29 March, 1996.

[19] D. Pisinger, P. Toth, "Knapsack Problems", in D.Z. Du, P. Pardalos (eds.) *Handbook of Combinatorial Optimization* (1998), Kluwer, Norwell, 1–89.

[20] M.E. Posner, M. Guignard, "The collapsing 0-1 knapsack problem", *Mathematical Programming* **15** (1978), 155-161.

[21] S. Sahni, "Approximate Algorithms for the 0/1 Knapsack Problem", *Journal of the ACM* **22** (1975), 115–124.

[22] F. Vanderbeck, "Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems", TR-14/96, University of Cambridge, 1996.