Adding $e_{14} + e_{23}$ to both sides of the relation (4.17) we get

$$\sum_{i=1}^{4} e_{ii} + e_{12} + e_{34} + e_{14} + e_{23} \leq 2e_{23} + 2e_{14} + e_{13} + e_{24}. \tag{4.18}$$

Now we obtain

$$\begin{aligned} cost(X_1, X_2) &= e_{12} + e_{34} + e_{14} + e_{23} \\ &\leq \sum_{i=1}^{4} e_{ii} + e_{12} + e_{14} + e_{34} + e_{23} \\ &\underset{(4.18)}{\leq} 2e_{23} + 2e_{14} + e_{13} + e_{24} \\ &\leq 2(e_{23} + e_{14} + e_{13} + e_{24}) = 2 \cdot cost(Y_1, Y_2) \end{aligned}$$

The above calculation directly implies

$$cost(X_1, X_2) \leq 2 \cdot cost(Y_1, Y_2) - \sum_{i=1}^{4} e_{ii} - e_{13} - e_{24}.$$

□

Exercise 4.3.3.4 Show that one can even prove $cost(X_1, X_2) < 2 \cdot cost(Y_1, Y_2)$ in the proof of Theorem 4.3.3.3. □

Exercise 4.3.3.5 Find input instances for which the approximation ratio of the solutions computed by Algorithm 4.3.3.1 is almost 2. □

Exercise 4.3.3.6 Let k-maximum cut problem (k-MAX-CUT) be a restricted version of MAX-CUT, where the degree of the input graphs is bounded by $k \geq 3$. Design a δ_k-approximation algorithm for k-MAX-CUT for every $k \geq 3$ in such a way that $\delta_k < \delta_{k+1} < 2$ for every $k \geq 3$. □

Summary of Section 4.3.3

There are hard optimization problems whose local optima (according to a reasonable structure of the space of feasible solutions) have costs that well approximate the cost of total optima. In such cases local search algorithms may provide a very efficient solution to the given optimization problem.

4.3.4 Knapsack Problem and PTAS

The knapsack problem (KP) is a representative of the class NPO(I) and so KP is one of the easiest NP-hard optimization problems according to the polynomial-time approximability. We use this problem to illustrate a few concepts for the

design of approximation algorithms. First, we use a compromise between total search and greedy algorithms in order to design a PTAS for the simple knapsack problem (SKP). We show that this PTAS is stable according to some reasonable distance measure but that this stability does not suffice to create a PTAS for KP. Then, simply modifying this PTAS for SKP, we obtain a new, superstable PTAS for SKP that provides a PTAS for the general KP. Finally, we "merge" dynamic programming with a kind of approximation of input instances in order to get an FPTAS for KP.

For every input instance $w_1, \ldots, w_n, b$ of SKP, and every $T \subseteq \{1, \ldots, n\}$, remember that $cost(T) = \sum_{i \in T} w_i$. If $cost(T) \leq b$, then T is a feasible solution in $\mathcal{M}(w_1, w_2, \ldots, w_n, b)$.

First, we observe that the simple greedy algorithm for SKP is already a 2-approximation algorithm for SKP.

Algorithm 4.3.4.1

Input: Positive integers $w_1, w_2, \ldots, w_n, b$ for some $n \in \mathbb{N}$.

Step 1: Sort $w_1, w_2, \ldots, w_n$. For simplicity we may assume $w_1 \geq w_2 \geq \cdots \geq w_n$.

Step 2: $T := \emptyset$; $cost(T) := 0$;

Step 3: **for** $i = 1$ **to** n **do**

if $cost(T) + w_i < b$ **then**

do begin $T := T \cup \{i\}$;

$cost(T) := cost(T) + w_i$

end

Output: T.

Clearly, Algorithm 4.3.4.1 works in the time $O(n \log n)$ because Step 1 takes $O(n \log n)$ time, Step 2 takes $O(1)$ time, and Step 3 takes $O(n)$ time.

To see that Algorithm 4.3.4.1 is a 2-approximation algorithm for SKP it is sufficient to show that $cost(T) \geq b/2$ or T is an optimal solution. Without loss of generality one can assume $b \geq w_1 \geq w_2 \geq \cdots \geq w_n$. Let $j + 1$ be the smallest integer not in T (i.e., $\{1, 2, \ldots, j\} \subseteq T$}). Note, that $T \neq \emptyset$ (i.e., $j \geq 1$) because $w_1 \leq b$. If $j = 1$, then $w_1 + w_2 > b$. Since $w_1 \geq w_2$ it is obvious that $cost(T) = w_1 > \frac{b}{2}$. Thus, we may assume $j \geq 2$. In general, we have

$$cost(T) + w_{j+1} > b \geq Opt_{\mathrm{SKP}}(w_1, \ldots, w_n, b).$$

Since $w_1 \geq w_2 \geq \cdots \geq w_n$,

$$w_{j+1} \leq w_j \leq \frac{w_1 + w_2 + \cdots + w_j}{j} \leq \frac{b}{j}. \tag{4.19}$$

Thus, $cost(T) > b - w_{j+1} \geq b - \frac{b}{j} \geq b/2$ for every integer $j \geq 2$.

The inequality (4.19) is the kernel of the design idea of the following PTAS for SKP. If $T_{Opt} = \{i_1, i_2, \ldots, i_r\}$, $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_r}$, is an optimal solution and one is able to compute a solution T that contains the j largest weights $w_{i_1}, w_{i_2}, \ldots, w_{i_j}$ of T_{Opt} and $cost(T) + w_{i_{j+1}} > b$, then the difference

$$cost(T_{Opt}) - cost(T) \leq w_{i_{j+1}} \leq \frac{b}{i_{j+1} - 1} \leq \frac{b}{j}$$

is small relative to $cost(T_{Opt})$. To achieve an output T of at least this quality one considers all subsets of $\{1, \ldots, n\}$ of the cardinality at most j extended by the greedy approach, and outputs the best one of them. Obviously, the amount of work grows with j, but the relative error decreases with j.

Algorithm 4.3.4.2

Input: Positive integers $w_1, w_2, \ldots, w_n, b$ for some $n \in \mathbb{N}$, and a positive real number ε, $0 < \varepsilon < 1$.

Step 1: Sort $w_1, w_2, \ldots, w_n$. For simplicity, we assume $b \geq w_1 \geq w_2 \geq \cdots \geq w_n$.

Step 2: $k := \lceil 1/\varepsilon \rceil$.

Step 3: For every set $S \subseteq \{1, 2, \ldots, n\}$ with $|S| \leq k$ and $\sum_{i \in S} w_i \leq b$, extend S to S^* by using the greedy approach described in Step 3 of Algorithm 4.3.4.1.

{The sets S are created sequentially in the lexicographical order by backtracking, and the up-till-now best S^* is always saved.}

Output: A set S^* with the maximal $cost(S^*)$ among all sets created in Step 3.

Theorem 4.3.4.3 *Algorithm 4.3.4.2 is a* PTAS *for* SKP.

Proof. First, we consider the time complexity $Time(n)$ of Algorithm 4.3.4.2 for an input (I, ε), where $I = w_1, \ldots, w_n, b$. Step 1 can be executed in time $O(n \log n)$ and Step 2 in time $O(1)$. The number of sets $S \subseteq \{1, \ldots, n\}$ with $|S| \leq k$ is

$$\sum_{0 \leq i \leq k} \binom{n}{i} \leq \sum_{0 \leq i \leq k} n^i = \frac{n^{k+1} - 1}{n - 1} = O(n^k).$$

Since these sets can be "created" in lexicographical order, one can construct the next set from the previous one in time $O(1)$. To extend a set S to S^* by the greedy approach costs $O(n)$ time. Thus,

$$Time(n) \leq \left[\sum_{0 \leq i \leq k} \binom{n}{i} \right] \cdot O(n) = O(n^{k+1}) = O(n^{\lceil 1/\varepsilon \rceil + 1}).$$

Now, we show that the approximation ratio

$$R_{\text{Algorithm 4.3.4.2}}(I, \varepsilon) \leq 1 + \frac{1}{k} \leq 1 + \varepsilon$$

for every input $I = w_1, w_2, \ldots, w_n, b$ of SKP. Let $M = \{i_1, i_2, \ldots, i_p\}$, $i_1 < i_2 < \cdots < i_p$ be an optimal solution for I, i.e., $cost(M) = Opt_{\text{SKP}}(I)$. We distinguish two possibilities according to the relation between p and $k = \lceil 1/\varepsilon \rceil$.

If $p \leq k$, then Algorithm 4.3.4.2 considers the set M in Step 3 and so S^* is an optimal solution (i.e., the relative error is 0).

Let $p > k$. Algorithm 4.3.4.2 extends the set $P = \{i_1, i_2, \ldots, i_k\}$ containing the indices of the k greatest weights $w_1, w_2, \ldots, w_{i_k}$ involved in $cost(M) = w_{i_1} + \cdots + w_{i_p}$. If $P^* = M$, then we are ready. If $P^* \neq M$, then there exists $i_q \in M - P^*$ such that $i_q > i_k \geq k$ and

$$cost(P^*) + w_{i_q} > b \geq cost(M). \tag{4.20}$$

Since

$$w_{i_q} \leq \frac{w_{i_1} + w_{i_2} + \cdots + w_{i_k} + w_{i_q}}{k+1} \leq \frac{cost(M)}{k+1} \tag{4.21}$$

we obtain

$$\begin{aligned} R(I, \varepsilon) &= \frac{cost(M)}{cost(S^*)} \leq \frac{cost(M)}{cost(P^*)} \underset{(4.20)}{\leq} \frac{cost(M)}{cost(M) - w_{i_q}} \\ &\underset{(4.21)}{\leq} \frac{cost(M)}{cost(M) - \frac{cost(M)}{k+1}} = \frac{1}{1 - \frac{1}{k+1}} = \frac{k+1}{k} \\ &= 1 + \frac{1}{k} \leq 1 + \varepsilon. \end{aligned}$$

□

We observe that Algorithm 4.3.4.2 is consistent for KP. An input $w_1, \ldots, w_n$, b, $c_1, \ldots, c_n$ of KP is an input of SKP if $w_i = c_i$ for $i = 1, \ldots, n$, so, the relative difference between w_i and c_i seems to be a natural distance measure from the specification $w_i = c_i$. This is the reason for defining the distance function $DIST$ for any input w_1, w_2, ..., w_n, b, c_1, ..., c_n of KP as follows.

$$\begin{aligned} &DIST(w_1, \ldots, w_n, b, c_1, \ldots, c_n) = \\ &\quad \max\left\{ \max\left\{ \frac{c_i - w_i}{w_i} \;\middle|\; c_i \geq w_i,\ i \in \{1, \ldots, n\} \right\}, \right. \\ &\qquad \left. \max\left\{ \frac{w_i - c_i}{c_i} \;\middle|\; w_i \geq c_i,\ i \in \{1, \ldots, n\} \right\} \right\}. \end{aligned}$$

Let $KP_\delta = (\Sigma_I, \Sigma_O, L, Round_{\delta, DIST}(L_I), \mathcal{M}, cost, maximum)$ for any $\delta \in \mathbb{R}^+$. Now we show that Algorithm 4.3.4.2 is stable according to $DIST$ but this result does not imply the existence of a PTAS for KP_δ for any $\delta > 0$.

Let us consider $\{\text{ASKP}_\varepsilon\}_{\varepsilon > 0}$ as the collection of $(1 + \varepsilon)$-approximation algorithms determined by Algorithm 4.3.4.2.

Lemma 4.3.4.4 *For every $\varepsilon > 0$ and every $\delta > 0$, the algorithm* ASKP_ε *is an* $(1+\varepsilon+\delta(2+\delta)\cdot(1+\varepsilon))$*-approximation algorithm for* KP_δ.

Proof. Let $w_1 \geq w_2 \geq \cdots \geq w_n$ for an input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$, and let $k = \lceil 1/\varepsilon \rceil$. Let $U = \{i_1, i_2, \ldots, i_l\} \subseteq \{1, 2, \ldots, n\}$ be an optimal solution for I. If $l \leq k$, then ASKP_ε outputs an optimal solution with $cost(U)$ because ASKP_ε has considered U as a candidate for the output in Step 3.

Consider the case $l > k$. ASKP_ε has considered the greedy extension of $T = \{i_1, i_2, \ldots, i_k\}$ in Step 2. Let $T^* = \{i_1, i_2, \ldots, i_k, j_{k+1}, \ldots, j_{k+r}\}$ be the greedy extension of T. Obviously, it is sufficient to show that the difference $cost(U) - cost(T^*)$ is small relative to $cost(U)$, because the cost of the output of SKP_ε is at least $cost(T^*)$. We distinguish the following two possibilities according to the weights of the feasible solutions U and T^*.

(i) Let $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j \leq 0$.

Obviously, for every i, $(1+\delta)^{-1} \leq \frac{c_i}{w_i} \leq 1+\delta$. Thus,

$$cost(U) = \sum_{i \in U} c_i \leq (1+\delta) \cdot \sum_{i \in U} w_i$$

and

$$cost(T^*) = \sum_{j \in T^*} c_j \geq (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j.$$

In this way we obtain

$$\begin{aligned}
cost(U) - cost(T^*) &\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j \\
&\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i \\
&= \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} w_i \\
&\leq \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} (1+\delta) \cdot c_i \\
&= \delta \cdot (2+\delta) \cdot \sum_{i \in U} c_i \\
&= \delta \cdot (2+\delta) \cdot cost(U).
\end{aligned}$$

Finally,

$$\frac{cost(U) - cost(T^*)}{cost(U)} \leq \frac{\delta \cdot (2+\delta) \cdot cost(U)}{cost(U)} = \delta \cdot (2+\delta).$$

(ii) Let $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j > 0$.

Let c be the cost of the first part of U with the weight $\sum_{j \in T^*} w_j$. Then, in the same way as in (i), one can establish

$$\frac{c - cost(T^*)}{c} \leq \delta \cdot (2 + \delta). \tag{4.22}$$

It remains to bound $cost(U) - c$, i.e., the cost of the last part of U with the weight d. Obviously, $d \leq b - \sum_{j \in T^*} w_j \leq w_{i_r}$ for some $r > k$, $i_r \in U$ (if not, then ASKP_ε would add i_r to T^* in the greedy procedure). Since $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_l}$,

$$d \leq w_{i_r} \leq \frac{w_{i_1} + w_{i_2} + \cdots + w_{i_r}}{r} \leq \frac{\sum_{i \in U} w_i}{k+1} \leq \varepsilon \cdot \sum_{i \in U} w_i. \tag{4.23}$$

Since $cost(U) \leq c + d \cdot (1 + \delta)$ we obtain

$$\begin{aligned}
\frac{cost(U) - cost(T^*)}{cost(U)} &\leq \frac{c + d \cdot (1+\delta) - cost(T^*)}{cost(U)} \\
&\underset{(4.23)}{=} \frac{c - cost(T^*)}{cost(U)} + \frac{(1+\delta) \cdot \varepsilon \cdot \sum_{i \in U} w_i}{cost(U)} \\
&\underset{(4.22)}{\leq} \delta \cdot (2+\delta) + (1+\delta) \cdot \varepsilon \cdot (1+\delta) \\
&= 2\delta + \delta^2 + \varepsilon \cdot (1+\delta)^2 \\
&= \varepsilon + \delta \cdot (2+\delta) \cdot (1+\varepsilon).
\end{aligned}$$

□

Corollary 4.3.4.5 *The* PTAS *Algorithm 4.3.4.2 is stable according to DIST, but not superstable according to DIST.*

Proof. The first assertion directly follows from 4.3.4.4. To see that SKP is not superstable according to *DIST* it is sufficient to consider the following input:[11]

$$w_1, w_2, \ldots, w_m, u_1, u_2, \ldots, u_m, b, c_1, c_2, \ldots, c_{2m},$$

where $w_1 = w_2 = \cdots = w_m$, $u_1 = u_2 = \cdots = u_m$, $w_i = u_i + 1$ for $i = 1, \ldots, m$, $b = \sum_{i=1}^{m} w_i$, $c_1 = c_2 = \cdots = c_m = (1 - \delta)w_1$ and $c_{m+1} = c_{m+2} = \cdots = c_{2m} = (1 + \delta)u_1$. □

We see that the PTAS SKP is stable according to *DIST*, but this does not suffice to get a PTAS for KP_δ for any $\delta > 0$. This is because in the approximation ratio we have the additive factor $\delta \cdot (2 + \delta)$ that is independent of ε. In what

[11] Note that m should be chosen to be essentially larger than ε^{-1} for a given ε.

follows we change Algorithm 4.3.4.2 a little bit in such a way that we obtain a PTAS for every KP_δ, $\delta > 0$. The modification idea is very natural – to sort the input values according to the cost per one weight unit.

Algorithm 4.3.4.6 PTAS MOD-SKP

Input: Positive integers $w_1, w_2, \ldots, w_n, b, c_1, \ldots, c_n$ for some $n \in \mathbb{N}$, and some positive real number ε, $1 > \varepsilon > 0$.

Step 1: Sort $\frac{c_1}{w_1}, \frac{c_2}{w_2}, \ldots, \frac{c_n}{w_n}$. For simplicity we may assume $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ for $i = 1, \ldots, n-1$.

Step 2: Set $k = \lceil 1/\varepsilon \rceil$.

Step 3: The same as Step 3 of Algorithm 4.3.4.2, but the greedy procedure follows the ordering of the w_is of Step 1.

Output: The best T^* constructed in Step 3.

Let $\text{MOD-SKP}_\varepsilon$ denote the algorithm given by PTAS MOD-SKP for any fixed $\varepsilon > 0$.

Lemma 4.3.4.7 *For every ε, $1 > \varepsilon > 0$ and every $\delta \geq 0$, $\text{MOD-SKP}_\varepsilon$ is a $1 + \varepsilon \cdot (1+\delta)^2$-approximation algorithm for KP_δ.*

Proof. Let $U = \{i_1, i_2, \ldots, i_l\} \subseteq \{1, 2, \ldots, n\}$, where $w_{i_1} \leq w_{i_2} \leq \cdots \leq w_{i_l}$[12] be an optimal solution for the input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$.

If $l \leq k$, then $\text{MOD-SKP}_\varepsilon$ provides an optimal solution.

If $l > k$, then we consider a $T^* = \{i_1, i_2, \ldots, i_k, j_{k+1}, \ldots, j_{k+r}\}$ as a greedy extension of $T = \{i_1, i_2, \ldots, i_k\}$. Again, we distinguish two possibilities according to the sizes of $\sum_{i \in U} w_i$ and $\sum_{j \in T^*} w_j$.

(i) Let $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.

Now, we show that this is impossible because it contradicts the optimality of U. Both $cost(U)$ and $cost(T^*)$ contain $\sum_{s=1}^{k} c_{i_s}$. For the rest T^* contains the best choice of w_is according to the cost of one weight unit. The choice of U per one weight unit cannot be better. Thus, $cost(U) < cost(T^*)$.

(ii) Let $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j \geq 0$.

Because of the optimal choice of T^* according to the cost per one weight unit, the cost c of the first part of U with the weight $\sum_{j \in T^*} w_j$ is at most $cost(T^*)$, i.e.,

$$c - cost(T^*) \leq 0 \tag{4.24}$$

[12] Observe that at this place we consider the order according to the weights and not according to the cost per one weight unit.

Since U and T^* contain the same k indices $i_1, i_2, \ldots, i_k$, and $w_{i_1}, \ldots, w_{i_k}$ are the largest weights in both U and T^*, the same consideration as in the proof of Lemma 4.3.4.4 yields (see (4.23))

$$d \leq \varepsilon \cdot \sum_{i \in U} w_i, \text{ and } cost(U) \leq c + d \cdot (1 + \delta). \tag{4.25}$$

Thus,

$$\begin{aligned} \frac{cost(U) - cost(T^*)}{cost(U)} &\leq \frac{c + d \cdot (1+\delta) - cost(T^*)}{cost(U)} \\ &\underset{(4.24)}{\leq} \frac{d \cdot (1+\delta)}{cost(U)} \\ &\underset{(4.25)}{\leq} \varepsilon \cdot (1+\delta) \cdot \frac{\sum_{i \in U} w_i}{cost(U)} \\ &= \varepsilon \cdot (1+\delta)^2 \end{aligned}$$

□

We observe that the collection of MOD-SKP$_\varepsilon$ algorithms is a PTAS for every KP_δ with a constant $\delta \geq 0$ (independent of the size $2n+1$ of the input).

Theorem 4.3.4.8 MOD-SKP *is superstable according to DIST, and so Algorithm 4.3.4.6 is a* PTAS *for* KP.

To obtain an FPTAS for KP we consider a completely new approach. In Section 3.2 we presented Algorithm 3.2.2.2 (based on dynamic programming) for KP working in time $O(n \cdot Opt_{\mathrm{KP}}(I))$ for any input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$. The trouble is that $Opt_{\mathrm{KP}}(I)$ can be exponential in the input length and so the time complexity of Algorithm 3.2.2.2 has to be considered to be exponential. The idea of our FPTAS for KP is to "approximate" every input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$ by another input with $I' = w_1, w_2, \ldots, w_n, b, c'_1, \ldots, c'_n$ with $\sum_{i=1}^{n} c'_i$ polynomial in n, and then to apply Algorithm 3.2.2.2 to the input I' in order to get a feasible solution for I. If one wants to obtain $Opt_{\mathrm{KP}}(I')$ to be small in n, then one has to divide the input values $c_1, \ldots, c_n$ by a same large number d. Obviously, the efficiency increases with growing d, but the approximation ratio increases with growing d, too. So, d may be chosen dependent on the user preference.

Algorithm 4.3.4.9 FPTAS for KP

Input: $w_1, \ldots, w_n, b, c_1, \ldots, c_n \in \mathbb{N}$, $n \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$.

Step 1: $c_{max} := \max\{c_1, \ldots, c_n\}$;

$t := \left\lfloor \log_2 \frac{\varepsilon \cdot c_{max}}{(1+\varepsilon) \cdot n} \right\rfloor$;

Step 2: **for** $i = 1$ **to** n
do $c'_i := \lfloor c_i \cdot 2^{-t} \rfloor$.

Step 3: Compute an optimal solution T' for the input
$I' = w_1, \ldots, w_n, b, c'_1, \ldots, c'_n$ by Algorithm 3.2.2.2.

Output: T.

Theorem 4.3.4.10 *Algorithm 4.3.4.9 is an* FPTAS *for* KP.

Proof. First, we show that Algorithm 4.3.4.9 is an approximation scheme. Since the output T' (an optimal solution for I') is a feasible solution for I', and I and I' do not differ in the weights $w_1, \ldots, w_n, b$, T' is a feasible solution for I, too. Let T be an optimal solution for the original input I. Our goal is to show that

$$R(I) = \frac{cost(T, I)}{cost(T', I)} \leq 1 + \varepsilon.$$

We have:

$$\begin{aligned}
cost(T, I) &= \sum_{j \in T} c_j \\
&\geq \sum_{j \in T'} c_j = cost(T', I) \quad \{\text{because } T \text{ is optimal for } I \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{and } T' \text{ is feasible for } I\} \\
&\geq 2^t \cdot \sum_{j \in T'} c'_j \quad \{\text{follows from } c'_j = \lfloor c_j \cdot 2^{-t} \rfloor\} \\
&\geq 2^t \sum_{j \in T} c'_j \quad \{\text{because } T' \text{ is optimal for } I'\} \\
&= \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor \quad \{\text{because } c'_j = \lfloor c_j \cdot 2^{-t} \rfloor\} \\
&\geq \sum_{j \in T} 2^t \left(c_j \cdot 2^{-t} - 1 \right) \geq \left(\sum_{j \in T} c_j \right) - n \cdot 2^t = cost(T, I) - n \cdot 2^t.
\end{aligned}$$

We have, thus, proved

$$\begin{aligned}
cost(T, I) &\geq cost(T', I) \geq cost(T, I) - n \cdot 2^t, \text{ i.e.,} && (4.26) \\
0 &\leq cost(T, I) - cost(T', I) \leq n \cdot 2^t \\
&\leq n \cdot \frac{\varepsilon \cdot c_{max}}{(1 + \varepsilon) \cdot n} = \varepsilon \cdot \frac{c_{max}}{1 + \varepsilon}. && (4.27)
\end{aligned}$$

Since we may assume $cost(T, I) \geq c_{max}$ ($w_i \leq b$ for every $i = 1, \ldots, n$), we obtain from (4.27)

$$cost(T', I) \geq c_{max} - \varepsilon \cdot \frac{c_{max}}{1 + \varepsilon}. \qquad (4.28)$$

Finally,

$$\begin{aligned} R(I) &= \frac{cost(T,I)}{cost(T',I)} = \frac{cost(T',I) + cost(T,I) - cost(T',I)}{cost(T',I)} \\ &\leq 1 + \frac{\varepsilon \cdot \frac{c_{max}}{1+\varepsilon}}{cost(T',I)} \quad \{\text{because of (4.27)}\} \\ &\leq 1 + \frac{\varepsilon \cdot \frac{c_{max}}{1+\varepsilon}}{c_{max} - \varepsilon \cdot \frac{c_{max}}{1+\varepsilon}} \quad \{\text{because of (4.28)}\} \\ &= 1 + \frac{\varepsilon}{1+\varepsilon} \cdot \frac{1}{1 - \frac{\varepsilon}{1+\varepsilon}} = 1 + \frac{\varepsilon}{1+\varepsilon} \cdot (1+\varepsilon) = 1 + \varepsilon. \end{aligned}$$

Now, we have to prove that the time complexity of Algorithm 4.3.4.9 is polynomial in n and ε^{-1}. Step 1 and Step 2 can be executed in time $O(n)$. Step 3 is the run of Algorithm 3.2.2.2 on the input I' and this can be performed in time $O(n \cdot Opt_{\mathrm{KP}}(I'))$. We bound $Opt_{\mathrm{KP}}(I')$ as follows:

$$\begin{aligned} Opt_{\mathrm{KP}}(I') &\leq \sum_{i=1}^{n} c'_i = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\varepsilon \cdot c_{max}}{(1+\varepsilon)\cdot n} \right\rfloor} \right\rfloor \\ &\leq \sum_{i=1}^{n} \left(c_i \cdot 2 \cdot \frac{(1+\varepsilon)\cdot n}{\varepsilon \cdot c_{max}} \right) \\ &= 2 \cdot (1+\varepsilon) \cdot \varepsilon^{-1} \cdot \frac{n}{c_{max}} \cdot \sum_{i=1}^{n} c_i \\ &\leq 2 \cdot (1+\varepsilon) \cdot \varepsilon^{-1} \cdot n^2 \in O\left(\varepsilon^{-1} \cdot n^2\right). \end{aligned}$$

Thus, Algorithm 4.3.4.9 works in time $O\left(\varepsilon^{-1} \cdot n^3\right)$. □

Exercise 4.3.4.11 Consider Algorithm 4.3.4.9 for parameters other than $d = 2^{-t}$. By a suitable choice of d find an FPTAS for KP, whose time complexity is quadratic in n.

Exercise 4.3.4.12 (*) Design a new FPTAS for KP such that the running time is in

(i) $O\left(n \cdot \log n + n \cdot \varepsilon^{-2}\right)$,

(ii) $O\left(n \cdot \log_2\left(\varepsilon^{-1}\right) + \varepsilon^{-4}\right)$.

Summary of Section 4.3.4

We presented the following two methods for the design of polynomial-time approximation algorithms:

(i) A combination of the greedy approach and total search in order to get a PTAS.

(ii) An application of a pseudo-polynomial-time optimization algorithm A that may be of complexity $O(p(n) \cdot F)$, where p is a polynomial in the number n of input variables and F is linear in the size of the values of the input variables. Since the value of a number is exponential in the size of its representation, F can be exponential in the input length. To be able to efficiently apply A, one "reduces" the inputs with F exponential to n to inputs with F polynomial to n. This may lead to a good approximation ratio for every input instance.

We have applied (i) to get a PTAS for SKP, and we have applied (ii) to design an FPTAS for KP. Moreover, we have shown that considering the notion of the superstability of approximation, one can find a natural way to extend the use of a PTAS to a class of problem instances that is essential larger than the original class.

4.3.5 Traveling Salesperson Problem and Stability of Approximation

The traveling salesperson problem (TSP) is a representative of the class NPO(V), i.e., one of the hardest optimization problems according to polynomial-time approximability.[13] Because of this one could consider the approximation approach unsuitable for TSP. We show here that this needs not be true even for the hardest problems from NPO(V). First, we show that there is a subset $L_\triangle$ of input instances such that

(i) one can design efficient approximation algorithms for inputs in $L_\triangle$,

(ii) several applications produce inputs either from $L_\triangle$ or inputs whose properties are not very far from the specification of $L_\triangle$.

This could already be useful for some applications but we still improve on it. We show that one can modify the above mentioned algorithms to get stable algorithms. In this way one can apply these algorithms for inputs outside $L_\triangle$ and efficiently compute an upper bound on the approximation ratio for every input.

First, we present two algorithms for $\triangle$-TSP, i.e., TSP with the inputs satisfying the triangle inequality.

Algorithm 4.3.5.1

Input: A complete graph $G = (V, E)$, and a cost function $c : E \to \mathbb{N}^+$ satisfying the triangle inequality

$$c(\{u, v\}) \leq c(\{u, w\}) + c(\{w, v\})$$

for all three different $u, v, w \in V$ {i.e., $(G, c) \in L_\triangle$}.

[13] More information about the inapproximability of TSP will be given in Section 4.4.