Technische Universität München, Zentrum Mathematik
Lehrstuhl für Angewandte Geometrie und Diskrete Mathematik

Discrete Optimization (MA 3502)

Prof. Dr. Nicole Megow | Dr. Fidaa Abed | Dr. Michael Ritter

Exercise Sheet 4

***Problem 4.1** (Knapsack Problem)
The 0-1-KNAPSACK-problem can be defined as follows:

GIVEN:	n goods with weights $a_1, \dots, a_n \in \mathbb{N}$, values $c_1, \dots, c_n \in \mathbb{N}$, and a weight-limit $b \in \mathbb{N}$. All goods are available only once.
TASK:	Find a collection of goods not exceeding the weight-limit b, of maximal value.

(Without loss of generality (Wlog) one may assume $b \geq \max_j a_j$.)

For $r \in [n] := \{1, \dots, n\}$ and $b' \in [b]$ define

$$z_r(b') := \max \left\{ \sum_{i=1}^{r} c_i x_i \mid \sum_{i=1}^{r} a_i x_i \leq b' \ , \ x_i \in \{0,1\} \ i \in [r] \right\}$$

Hence, our goal is to compute $z_n(b)$.

a) Show $z_n(b) = \max\{z_{n-1}(b), z_{n-1}(b - a_n) + c_n\}$.

b) Use (a) to construct an algorithms solving 0-1-KNAPSACK in time $O(b \cdot n)$.

c) Discuss the running time of the algorithm with respect to the definition of efficiency in complexity theory.

Answer to Problem 4.1

a) $z_r(b')$ denote the maximum value of the knapsack with weight-limit b', using the elements $a_1, \dots, a_r$.

The core of the ALGORITHM:
FOR b' FROM 1 TO b DO

$$z_1(b') := \begin{cases} 0 & \text{für } a_1 > b' \\ c_1 & \text{for } a_1 \leq b' \end{cases}$$

FOR r FROM 1 TO n DO
$z_r(b') := \max\{z_{r-1}(b'), z_{r-1}(b' - a_r) + c_r\}$
END FOR
END FOR

For the running time it is easy to see that the running time is $O(nb)$. Which is not polynomial of the input but pseudopolynomial due to the fact that the number b is represented in binary in the input.

Problem 4.2 ($\mathcal{NP}$-hardness of the knapsack problem)
Show that the knapsack problem is $\mathcal{NP}$-hard by giving a reduction of the $\mathcal{NP}$-hard *partition problem.*

Problem: Partition
Input: Set of natural numbers $A = \{a_1, \ldots, a_n\} \subset \mathbb{N}$
Question: Is there a partition $A_1, A_2 := A/A_1$ of A such that

$$\sum_{a \in A_1} a = \sum_{a \in A_2} a?$$

to the knapsack problem.

Answer to Problem 4.2
For an instance $A = \{a_1, \ldots, a_n\}$ of Partition, we construct an instance $(v_1, \ldots, v_n, w_1, \ldots, w_n, b, v)$ of knapsack with n goods, where the weights and values of the goods correspond to the numbers of A: $v_i := a_i =: w_i$. The bound on the weight and the value to achieve are set to $v := 1/2 \sum_{a \in A} a =: b$. There is a solution $I \subset \{1, \ldots, n\}$ to this knapsack instance with value $\geq v$ if and only if there is a partition of A: Given a valid parition of A, $I := \{i \in \{1, \ldots, n\} \mid a_i \in A\}$ is a feasible solution of the knapsack instance with value $v = 1/2 \sum_{a \in A} a = b$. Similarly, a solution I of the knapsack instance must have value $v = 1/2 \sum_{a \in A} a$ and therefore, $A_1 := \{a_i \in A \mid i \in I\}$ and $A_2 := A_1^c$ is a valid parition of A.

Problem 4.3 (Dynamic Programming for TSP)
Design an algorithm that solves the metric TSP in running time $\mathcal{O}\left(n^2 2^n\right)$ using a dynamic programming approach. Use the following notation and variables:

- $S \subset \{2, 3, \ldots, n\}$ and $s \in S$.
- bestpath$[S, s]$:= shortest path from 1 to s which passes through all vertices in $S \cup \{1\}$.
- cost$[S, s]$:= length of bestpath$[S, s]$.

Where in your algorithm / proof do you use the fact that the distance function is metric?

Answer to Problem 4.3
We will show that Algorithm 1 computes an optimal TSP tour in running time $\mathcal{O}\left(n^2 2^n\right)$. Let $G = (V, E)$ be a complete graph and $c : E \longrightarrow \mathbb{R}_{\geq 0}$ a metric distance function on G. At first, we verify that the computed tour is indeed optimal. Is is sufficient to show that cost$[S, s]$ satisfies the equation

$$\text{cost}[S, s] = \min_{k \in S \setminus \{s\}} \{\text{cost}[S \setminus \{s\}, k] + c(\{k, s\})\}. \tag{1}$$

Let $(1, x_1 \ldots, x_i, s)$ a shortest path from 1 to s through $S \cup \{1\}$. Then $(1, x_1, \ldots, x_i)$ is necessarily a shortest path from 1 to x_i through $(S \setminus \{s\}) \cup \{1\}$. The length of the path $(1, x_1 \ldots, x_i, s)$ is then given by cost$[S \setminus \{s\}, x_i] + c(\{x_i, s\})$.

For the determination for the running time, notice that for each subset S, we iterate over s and k. Since there are 2^n such subsets, the running time is $\mathcal{O}\left(n^2 2^n\right)$.

Notice that the triangle inequality is required during the initialization of the best path –bestpath$[\{i\}, i] := (1, i)$ – to ensure that there is no shorter path from 1 to i. But if we define bestpath$[S, s]$ to be the shortest path from 1 to s which passes through all vertices in $S \cup \{1\}$ and does not use any other

Input : Complete Graph G on n vertices, metric cost function c
Output : optimal TSP tour besttour

for $i := 2$ *to* n **do**
  $\text{cost}[\{i\}, i] := c(\{1, i\})$
  $\text{bestpath}[\{i\}, i] := (1, i)$
for $j := 2$ *to* n **do**
  foreach $S \subset \{2, 3, \ldots, n\}$ *with* $|S| = j$ **do**
    foreach $s \in S$ **do**
      $\text{cost}[S, s] := \min_{k \in S \setminus \{s\}} \{\text{cost}[S \setminus \{s\}, k] + c(\{k, s\})\}$
      $k := \text{argmin}_{k \in S \setminus \{s\}} \{\text{cost}[S \setminus \{s\}, k] + c(\{k, s\})\}$
      $\text{bestpath}[S, s] := (\text{bestpath}[S \setminus \{s\}, k], s)$
$\text{mincost} := \min_{k \neq 1} \{\text{cost}[\{2, 3, \ldots, n\}, k] + c(\{k, 1\})\}$
$k := \text{argmin}_{k \neq 1} \{\text{cost}[\{2, 3, \ldots, n\}, k] + c(\{k, 1\})\}$
$\text{besttour} := \text{bestpath}[\{2, 3, \ldots, n\}, k]$

Algorithm 1: Dynamic Programming for TSP

vertices, the proof for the correctness of Equation (1) is still correct. Therefore the algorithm also computes correct solutions for non-metric TSP instances.

***Problem 4.4** (The Chvátal closure)
Given the following polyhedron P.

$$-2x_1 + x_2 \leq 0$$
$$2x_1 + x_2 \leq 6$$
$$-x_2 \leq -1$$

a) Show that the convex hull conv{(1,1),(2,1),(1,2),(2,2),(1.5,2.5)} is the (first) *Chvátal closure* of the P.

b) Prove that the inequality $x_2 \leq 2$ can not be derived from the inequalities describing P.

c) Show that P has Chvatal rank 2.

Answer to Problem 4.4

a) The (first) *Chvátal closure* of the P is the following:

By choosing $y^T = (0, 1/2, 1/2)$ we can get $x_1 \leq 5/2$. By rounding we get $x_1 \leq 2$.
By choosing $y^T = (1/2, 0, 1/2)$ we can get $-x_1 \leq -1/2$. By rounding we get $-x_1 \leq -1$.
By choosing $y^T = (5/6, 1/3, 1/6)$ we can get $-x_1 + x_2 \leq 1$. By rounding we get $-x_1 + x_2 \leq 1$.
By choosing $y^T = (1/3, 5/6, 1/6)$ we can get $x_1 + x_2 \leq 29/6$. By rounding we get $x_1 + x_2 \leq 4$.

b) Assume that the inequality $x_2 \leq 2$ can be derived from the inequalities describing P. Let $y^T Ax \leq \left\lfloor y^T b \right\rfloor$. This gives $(-2y_1 + 2y_2)x_1 + (y_1 + y_2 - y_3)x_2 \leq \lfloor 6y_2 - y_3 \rfloor$. Recall that $1 \leq x_1$ then $y_1 = y_2$. Also $y_3 = 2y_2 - 1$ and $\lfloor 6y_2 - (2y_2 - 1) \rfloor = \lfloor 4y_2 + 1) \rfloor = 2$. So $4y_2 < 2$ and $y_3 < 0$ which is a contradiction.

c) We can get $x_2 \leq 2$ from $x_1 + x_2 \leq 4$ and $-x_1 + x_2 \leq 1$ by choosing $y^T = (1/2, 1/2)$. After adding this cut we get the integral polyhedron. So the Chvatal rank of P is 2.