Combinatorial Optimization 2 Summer Term 2018

1. In the lecture the minimum weight set cover problem was introduced.

   (a) Show that the minimum weight vertex cover problem (given an undirected graph $G = (V, E)$ and nonnegative vertex weights w_v, find a vertex cover $V' \subseteq V$ of minimum weight $w(V') = \sum_{v \in V'} w_v$) is a special case of the minimum weight set cover problem.
   Describe the algorithm that results results from specializing the greedy algorithm discussed in the lecture to the vertex cover problem (find a vertex cover of minimum cardinality)?

   (b) Show that the minimum weight edge cover problem (given an undirected graph $G = (V, E)$ and nonnegative edge weights w_e, find an edge cover $E' \subseteq E$ of minimum weight $w(E') = \sum_{e \in E'} w_e$) is a special case of the minimum weight set cover problem.

2. Consider the following instance of the minimum weight set cover problem: $U = \{1, 2, 3, 4\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$ where $S_1 = \{1, 2, 4\}$, $S_2 = \{3\}$, $S_3 = \{1, 3\}$, $S_4 = \{4\}$, $S_5 = \{1, 2\}$. The costs $c_i = c(S_i)$ are as follows: $c_1 = 9$, $c_2 = 4$, $c_3 = 7$, $c_4 = 2$, $c_5 = 8$.

   (a) Apply the Greedy algorithm.

   (b) What lower bounds are obtained from this heuristic solution? Can you derive stronger lower bounds for this instance.

   (c) Apply the algorithm by Bar-Yehuda and Even.

3. Consider the instance of the vertex cover problem that results from the bipartite graph $G_q = (V, E)$ with $V = L \cup R$ obtained as follows: L is a set of q vertices. Next, for $i = 2, \ldots, q$, we add a set R_i of $\lfloor q/i \rfloor$ vertices to R, each one of them of degree i such that all of them (i.e. all vertices of degree i in R) are connected to distinct vertices in L.

   (a) Prove that the greedy algorithm returns the set R as vertex cover.

   (b) Show that the resulting approximation ratio is $\Omega(\log n)$ where n denotes the number of vertices of G_q.

   (c) Apply Gavril's 2-approximation algorithm to obtain a vertex cover for G_q. Which approximation ratio results for this specific instance?

   (Comment: This example shows that the greedy algorithm's approximation ratio stays logarithmic in the worst case for the vertex cover special case and does not improve as in the edge cover case.)

4. In the *shortest superstring problem* (SSP) we are given a finite alphabet Σ and a set of n strings, $S = \{s_1, \ldots, s_n\}$ over Σ (where we assume w.l.o.g. that no string s_i is a substring of another string s_j, $j \neq i$). The task is to find a shortest string s that contains each s_i as substring.

   The SSP has several applications (e.g. in computational biology) and is known to be NP-hard. The best known approximation algorithm has a performance ratio of 2.5.

   (a) Consider the following simple greedy algorithm for SSP: We define the overlap of two strings s and t to be the maximum length of a suffix of s that is also a prefix of t. The algorithm maintains a set of strings T which is initialized to S. At each step the algorithm chooses two strings from T with maximum overlap and replaces them with the string obtained by overlapping the two strings as much as possible. After $n - 1$ steps T will contain a single string and we are finished.
   Provide an example that demonstrates that this algorithm cannot lead to a better approximation factor than 2. (Comment: It's unknown whether there exist instances with a performance worse than 2.)

(b) Consider the following reduction to the minimum set cover problem. For strings s_i and s_j and $k > 0$ such that the last k symbols of s_i are the same as the first k symbols of s_j, let σ_{ijk} be the string obtained by overlapping these k positions of s_i and s_j. Let S' be the set of strings σ_{ijk} for all valid choices of i, j, k. For a string π let $\mathrm{set}(\pi) = \{s \in S \mid s \text{ is a substring of } \pi\}$. Now consider the instance of the minimum set cover problem with ground set $U = S$, setsystem $\mathcal{F} = \{\mathrm{set}(\pi) \mid \pi \in S \cup S'\}$ and costs $c(F) = |\pi|$ for $F \in \mathcal{F}$ with $F = \mathrm{set}(\pi)$ where $|\pi|$ denotes the length of string π.

Apply the greedy set cover algorithm to this instance and let $\mathrm{set}(\pi_1), \ldots, \mathrm{set}(\pi_k)\}$ be the sets in the picked cover. Then concatenate the strings $\pi_1, \ldots, \pi_k$ in any order to obtain a solution s for the SSP.

Prove that this provides a $2H_n$-approximation algorithm for SSP.

5. Let an instance of the knapsack problem with n items, profits $c_1, \ldots, c_n$, weights $w_1, \ldots, w_n$ and knapsack capacity W be given. We assume that profits and weights are positive integers and that the items are sorted such that
$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n}.$$
Let k be the critical item, i.e., $k := \min\{j \in \{1, \ldots, n\} : \sum_{i=1}^{j} w_i > W\}$. Show that x given by
$$x_j := \begin{cases} 1 & \text{for } j = 1, \ldots, k-1 \\ \frac{W - \sum_{i=1}^{k-1} w_i}{w_k} & \text{for } j = k \\ 0 & \text{for } j = k+1, \ldots, n \end{cases}$$
provides an optimal solution of the LP-relaxation of the knapsack problem.

6. Prove that the assumption that the profits c_j and the weights w_j in the knapsack problem are positive can me made without loss of generality.

7. Consider again the set-up of Problem 4. Prove that the following variation of the greedy algorithm for the knapsack problem is a 2-approximation algorithm.

   Let J_1 be the packing which packs the items $\{1, \ldots, k-1\}$ and let J_2 be the packing that packs an item with highest profit c_j into the knapsack. Output the better solution of the two.

8. Let an instance of the knapsack problem be given. Denote by z^* its optimal objective function value and denote by z^R the optimal objective function of the associated LP-relaxation.

   (a) Prove that $z^R \leq 2z^*$.

   (b) Show that there are knapsack instances such that $z^R \geq 2z^* - \varepsilon$ for every $\varepsilon > 0$ (i.e., the bound in (a) is asymptotically tight).

9. Provide an example that shows there does not exist a constant k such that the (unmodified) greedy algorithm for the knapsack algorithmis is a k-approximation algorithm.

10. Provide the recursion for the DP2 variant of dynamic programming for the knapsack problem we discussed in class.

11. Consider the following instance of the knapsack problem with $n = 8$, $W = 8$ and

j	1	2	3	4	5	6	7	8
c_j	2	4	1	3	2	3	7	3
w_j	5	3	1	2	1	7	4	3

   (a) Solve this instance by applying the $O(nW)$ time dynamic programming approach DP1.

(b) Solve this instance by applying the $O(nU)$ time dynamic programming approach DP2.

12. Devise a pseudopolynomial time algorithm for the following problem:

    Given: Nonnegative integers n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ and W
    Task: Find a subset $J \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in J} w_j \geq W$ and $\sum_{j \in J} c_j$ is minimized.

13. In the multiple knapsack problem (MKP) we are given n items with item j having profit c_j and weight w_j. We are given m knapsacks with capacities $W_1, W_2, \ldots, W_m$. The goal is to find a subset J of items of maximum total profit such that they can be packed into the given set of knapsacks (each selected item is packed into exactly one knapsack).

    (a) Formulate the (MKP) as an integer program.
    (b) Provide a dynamic programming algorithm for the MKP with 2 knapsacks with running time $O(nW_1W_2)$.
    (c) Extend your approach in (b) to the case of m knapsacks.
    (d) Let $m = 2$. Can you get a pseudopolynomial time algorithm with a running time which is polynomial in n and $P = \sum_{j=1}^{n} p_j$?
    (e) Argue that there exists no FPTAS for the MKP with 2 knapsacks.

14. Consider the following linear program

$$\begin{aligned}
\max \sum_{i=1}^{m} \sum_{j=1}^{n} c_j x_{ij} & & \\
\text{s.t.} \sum_{j=1}^{n} w_j x_{ij} \leq W_i & \quad & \text{for } i = 1, \ldots, m \\
\sum_{i=1}^{m} x_{ij} \leq 1 & & \text{for } j = 1, \ldots, n \\
0 \leq x_{ij} \leq 1 & & \text{for } i = 1, \ldots, m;\ j = 1, \ldots, n.
\end{aligned}$$

    (a) Argue that this LP can be seen as the LP-relaxation of a variant of the multi knapsack problem. What is the meaning of the now fractional x_{ij} variables? What is the difference to the MKP as introduced in Problem 12?
    (b) Show that the LP can be solved in polynomial time by a combinatorial approach (hint: flow problem).

15. Solve the following instance of the knapsack problem

$$\max\ 7x_1 + 16x_2 + 11x_3 + 14x_4 + 24x_5 + 60x_6$$

    s.t.

$$3x_1 + 6x_2 + 4x_3 + 5x_4 + 8x_5 + 2x_6 \leq 16, x_i \in \{0, 1\}$$

    by the branch&bound approach.

16. Apply the bin packing heuristics/approximation algorithms discussed in the lecture to the following instance: $n = 7$, $a_1 = 0.2$, $a_2 = 0.5$, $a_3 = 0.4$, $a_4 = 0.7$, $a_5 = 0.1$, $a_6 = 0.3$, $a_7 = 0.8$.

17. Show that the bin packing problem can be solved in polynomial time for instances where $a_j > \frac{1}{3}$ holds for all $j = 1, \ldots, n$.

18. Prove that there cannot be an online approximation algorithm for the bin packing problem with a performance ratio $< \frac{4}{3}$ unless P$\neq$NP. (Hint: Consider a list with n items such the first half of the items has size $\frac{1}{2} - \varepsilon$ and the second half has size $\frac{1}{2} + \varepsilon$.)

19. Find a bin packing instance I such that $\text{FF}(I) = 17$ und $\text{OPT}(I) = 10$ holds.

20. Consider the following class of bin packing instances I and apply the First Fit decreasing (FFD) heuristic: Let $\varepsilon > 0$ be sufficiently small and let $m \in \mathbb{N}$. There are $n = 30m$ items with the following sizes: The first $6m$ items have size $\frac{1}{2} + \varepsilon$, the next $6m$ items have size $\frac{1}{4} + 2\epsilon$, the next $6m$ items have size $\frac{1}{4} + \varepsilon$ and the remaining $12m$ items have size $\frac{1}{4} - 2\varepsilon$. Compute $\text{FFD}(I)$ and $\text{OPT}(I)$.

21. Consider two bin packing instances I_1 with items a_i, $i = 1, \ldots, p$ and I_2 with items a'_i, $i = 1, \ldots, q$. We say that I_2 covers I_1 if there are indices $1 \leq j_1 < j_2 < \cdots < j_p \leq q$ with $a_i = a'_{j_i}$ for $i = 1, \ldots, p$. An algorithm for the bin packing problem is called monotone if for instances I_1 and I_2 such that I_2 covers I_1 the algorithm needs at least as many bins for I_2 than for I_1.

    (a) Show that NEXT FIT is monotone.
    (b) Show that FIRST FIT is not monotone.
    (c) Is FIRST FIT DECREASING monotone?

22. We are given an instance of the bin packing problem and a fixed number $k \in \mathbb{N}$. Devise a pseudopolynomial time algorithm that decides if there exists a solution for this instance which uses at most k bins.

23. Provide an algorithm for the bin packing problem with a constant number m of different item sizes that runs in time polynomial in n the number of items. (Hint: Dynamic programming.)

24. Prove that the second step in the algorithm of Fernandez de la Vega and Luecker can be implemented to run in $O(n \log \frac{1}{\varepsilon})$ time.

25. Consider the following instance of the bin packing problem in compact form: We are given the following 4 item sizes $s_1 = 45/100$, $s_2 = 36/100$, $s_3 = 31/100$ and $s_4 = 14/100$. The multiplicities are as follows: $b_1 = 97$, $b_2 = 610$, $b_3 = 395$, $b_4 = 211$.

    (a) Apply the FFD-heuristic to this instance.
    (b) Solve the LP-relaxation of the compact integer programming model introduced in the lecture. Which bound does it provide for the BP instance?

26. Prove the bound $\lceil (SUM(I)/(1-\gamma) \rceil$ stated in the lecture for the approximation ratio of NEXT FIT for instances where all item sizes s_i fulfill $s_i < \gamma$ where $0 < \gamma < 1$.

27. Consider the following scheduling problem: We are given m machines and a set of n jobs along with a processing time p_j for each job j, $j = 1, \ldots, n$. The goal is to find an assignment of the jobs to the machines (each jobs need to be executed on exactly one machine) such that the makespan (maximum of the completion times over all jobs) is minimized.

    (a) Prove that the problem is NP-hard (for ambitioned students: strongly NP-hard).
    (b) Prove that the greedy algorithm that goes through the job sequence in arbitrary order and assigns the current job to the machine which is the least busy one up to that point (if there are several such jobs, break the tie arbitrarily) is a 2-approximation algorithm.
    (c) Obtain an FPTAS for this problem *for fixed* m (i.e. m is considered as a constant, and not as part of the input).