Optimization (2MMD10/2DME20) TU/e, fall 2015
Problem set for weeks $\boxed{\text{4 and 5}}$

Exercise 69. State for each of the following decision problems whether you think that it is contained in NP and/or co-NP. If you claim containment, then provide a corresponding certificate. If you claim non-containment, then state your intuition about it.

(a) Instance: A logical formula F in CNF. Question: Does F possess at most one satisfying truth assignment?

(b) Instance: A logical formula F in CNF. Question: Does F possess a unique satisfying truth assignment?

(c) Instance: A logical formula F in CNF. Question: Is there a truth assignment for which F evaluates to FALSE?

(d) Instance: A set of n cities; the distances between these cities; a bound B. Question: Is the length of the shortest TSP-tour equal to B?

(e) Instance: An edge-weighted, undirected graph G; a bound B. Question: Does G have a spanning tree of weight at least B?

(f) Instance: An edge-weighted, undirected graph G. Question: Is the minimum weight perfect matching in G unique?

Exercise 70. Consider a decision problem X, and assume that every instance of X can be rewritten in polynomial time into an equivalent instance of SAT.
(a) True or false: Then X lies in NP.
(b) True or false: Then X is NP-hard.

Exercise 71. We saw in the course that 3-SAT is NP-hard. Which of the following variants of SAT are polynomially solvable?
(a) Every clause contains exactly two literals
(b) Every clause contains at most one un-negated literal
(c) Every clause contains at most one negated literal
(d) Every variable occurs in at most two clauses

Exercise 72. ODD-3SAT asks whether a given logical formula Φ in 3-CNF over variable set X possesses a truth setting for X such that every clause contains an odd number of true literals. Decide whether ODD-3SAT can be solved in polynomial time.

Exercise 73. A set of closed intervals on the real line is called *independent*, if no two of these intervals share a common point. The problem INDEPENDENT INTERVAL SET asks for a given set S of intervals and a given bound k, whether there exists an independent subset of at least k intervals.
(a) Decide whether INDEPENDENT INTERVAL SET $\leq_p$ INDEPENDENT SET.
(b) Decide whether INDEPENDENT SET $\leq_p$ INDEPENDENT INTERVAL SET.

Exercise 74. Prove NP-hardness of the PARTITION problem. An instance consists of positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$. The question is whether there exists an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$.

Exercise 75. Prove NP-hardness of the ZERO-CYCLE problem. An instance consists of an undirected graph $G = (V, E)$ together with a weight function $w : E \to \mathbb{Z}$. The question is whether the graph contains a simple cycle so that the sum of edge weights along the cycle equals zero.

Exercise 76. Prove NP-hardness of the HAMILTONIAN CYCLE problem in undirected bipartite graphs.

Exercise 77. An instance of the HAMILTONIAN PATH problem consists of a graph $G = (V, E)$ and two vertices $s.t \in V$. The question is to decide whether the graph contains a simple path connecting s to t and visiting every vertex in the graph exactly once.

Prove NP-hardness of the HAMILTONIAN PATH problem in (a) undirected graphs; (b) directed graphs.

Exercise 78. An instance of the DEGREE CONSTRAINED SPANNING TREE problem consists of an undirected graph $G = (V, E)$ and a positive integer d, and asks whether G has a spanning tree with maximum degree bounded by d. Prove NP-hardness of the DEGREE CONSTRAINED SPANNING TREE problem.

Exercise 79. Prove NP-hardness of the SET-COVER problem. An instance consists of a ground-set X, subsets $S_1, \ldots, S_m$ of X, and an integer k. The question is whether there exists a sub-collection of k of the given subsets whose union is X?

Exercise 80. Consider the single clause $c = (x \vee y \vee z)$ and the following set S_c of ten clauses:

$$(x),\ (y),\ (z),\ (w);\quad (\bar{x} \vee \bar{y}),\ (\bar{x} \vee \bar{z}),\ (\bar{y} \vee \bar{z});\quad (x \vee \bar{w}),\ (y \vee \bar{w}),\ (z \vee \bar{w})$$

(a) Show: If a truth-setting for x, y, z satisfies the clause c, then it can be extended to a truth-setting for w such that seven clauses in S_c are satisfied.

(b) Show: If a truth-setting for x, y, z fails to satisfy clause c, then every extension to w satisfies at most six clauses in S_c.

(c) Deduce (by a reduction from 3-SAT) the NP-hardness of the following problem: Given a 2-SAT formula (where every clause contains at most two literals) and an integer k, decide whether there is a truth-setting that satisfies at least k clauses.

Exercise 81. A graph possesses a 3-coloring, if its vertices can be colored with three colors (red, blue, yellow) such that adjacent vertices always receive different colors.

Consider the following nine-vertex graph G_9. Two triangles y_1, y_2, y_3 and y_4, y_5, y_6 are connected to each other by the edge $[y_1, y_4]$. Furthermore there are three vertices a, b, c and the three edges $[a, y_2]$, $[b, y_3]$ and $[c, y_5]$. We consider 3-colorings of the vertices of G_9.

(a) Prove: If a, b, c are all colored by the same color f, then also y_6 must be colored f.

(b) Prove: Any coloring of a, b, c that assigns color f to at least one of a, b, c can be extended to a coloring of G_9 that assigns color f to vertex y_6.

(c) Deduce (by a reduction from 3-SAT): Deciding whether a given input graph has a 3-coloring is NP-complete.

Exercise 82. A single machine has to process n jobs $j = 1, \dots, n$; job j has an integer processing time of p_j time units, and should be completed by its due date d_j. A job is said to be *late* if it is completed after its due date; otherwise it is *early.* A fundamental problem in scheduling theory is to schedule the jobs so as to minimize the *number of late jobs.* The Moore-Hodgson algorithm solves this problem in $O(n \log n)$ time: Schedule the jobs in order of non-decreasing due dates; as soon as a scheduled job is late, remove the job with the largest processing time from the schedule, and mark it to be late. We consider three generalizations of this problem.

(a) Each job j ($j = 1, \dots, n$) has a given weight $w_j \in \mathbb{Z}^+$, and we want to minimize the *weighted number of late jobs*, that is, the sum of the weights of the late jobs. Prove that this problem is NP-hard.

(b) Each job j has a release date r_j (and hence cannot be processed before r_j). Prove that it is NP-hard to minimize the *number of late jobs.*

(c) Given is an acyclic directed graph G with the jobs as vertices. If G contains a directed path from vertex j to vertex k, then job k cannot be started before job j has been completed. We now want to minimize the *number of late jobs* subject to these *precedence constraints.* Prove that this problem is NP-hard.

Exercise 83. You are helping to organize a summer sports camp that covers n different sports (soccer, volleyball, etc). For each sport, the camp is supposed to have at least one councelor who is skilled in that sport. You have received application letters from m potential councelors that specify their skills. Your goal is to hire the smallest possible number of councelors so that all n sports are covered.

Formulate the resulting optimization problem as a decision problem. Prove that the decision problem is NP-complete.

Exercise 84. Harry Potter is looking for a Bowtruckle that has made itself invisible and is hiding in one of the vertices of a graph. Harry Potter repeatedly points his magic wand at some vertex and casts the SEMI-REVELIO spell. When the spell hits the Bowtruckle for the first time, the Bowtruckle leaves its vertex and moves to an adjacent vertex; as the Bowtruckle moves silently and invisibly, Harry is not aware of the move. When the spell hits the Bowtruckle for the second time, it finally becomes visibile to Harry.

Harry wants to make the Bowtruckle visible while casting as few SEMI-REVELIO spells as possible. A vertex sequence is called *revealing*, if casting the SEMI-REVELIO spells at the vertices according to the sequence guarantees that the Bowtruckle eventually becomes visibile (independently of its initial hiding place).

Problem: REVEALING SEQUENCE
Instance: A graph $G = (V, E)$; a bound k
Question: Does there exist a revealing vertex sequence of length at most k?

(a) Prove that REVEALING SEQUENCE lies in NP.
(b) Prove that REVEALING SEQUENCE is NP-hard.

Exercise 85. *"A peasant had to transport to the far side of a river a wolf, a goat, and a bundle of cabbages. The only boat he could find was one which would carry only two of them. For that reason he sought a plan which would enable them all to get to the far side unhurt. Let him, who is able, say how it could be possible to transport them safely?"* In a safe transportation plan, neither wolf & goat nor goat & cabbage can be left alone together. There exists a solution where the peasant makes seven boat trips across the river.

Consider the following generalization to arbitrary graphs $G = (V, E)$: Now the peasant has to transport a set V of items/vertices across the river. Two items are connected by an edge in E, if they are *conflicting* and thus cannot be left alone together without human supervision. The available boat has capacity $b \geq 1$, and thus can carry the man together with any subset $S \subseteq V$ of at most b items. The question is whether there exists a safe transportation plan that allows the peasant to get all of V safely to the other side.

Problem: Feasible Transportation Plan
Instance: A graph $G = (V, E)$; a boat of capacity $b \geq 1$.
Question: Does there exist a safe transportation plan for G and b?

Problem: Short Transportation Plan
Instance: A graph $G = (V, E)$; a boat of capacity $b \geq 1$.
Question: Does there exist a safe transportation plan for G and b, in which the peasant only makes three boat trips (one forward, one back, one forward)?

(a) Prove that FEASIBLE TRANSPORTATION PLAN is NP-hard.
(b) Prove that problem SHORT TRANSPORTATION PLAN is NP-hard.
(c) Prove that SHORT TRANSPORTATION PLAN lies in NP.

Exercise 86. In the SUBSET SUM problem, we are given positive integers $a_1, \ldots, a_n$ and b. The problem is to decide whether there exists an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = b$. We introduce a Boolean array $A[0 \ldots n, 0 \ldots b]$, and we set the entry $A[m, c]$ to TRUE if and only if there exists an index set $I \subseteq \{1, \ldots, m\}$ with $\sum_{i \in I} a_i = c$.
(a) Show that the array entries can be computed in overall time $O(nb)$.
(b) Deduce that SUBSET SUM can be solved in time $O(nb)$.
(c) Does this result imply P=NP?

Exercise 87. Assume that some decision problem X has a solution algorithm that uses polynomial time to recognize NO-instances, but uses exponential time to recognize YES-instances. Show that X lies in P.

Exercise 88. Assume that P=NP holds.
(a) Show that there exists a polynomial time algorithm that constructs a satisfying truth assignment for YES-instances of SAT.
(b) Show that there exists a polynomial time algorithm that constructs a Hamiltonian cycle for YES-instances of HAMILTONIAN CYCLE.

Exercise 89. Suppose that there is a black box that takes as input an undirected graph $G = (V, E)$ and an integer bound k, and then behaves as follows:

- If G is not connected, then the box returns "Not connected".
- If G is connected and contains an independent set of size k, the box returns "NO!".
- If G is connected and does not contain any independent set of size k, it returns "YES!".

The box always finds its answer in polynomial time (measured in the size of G and k). Show that with such a box you can solve the INDEPENDENT SET problem in polynomial time.

Exercise 90. Consider the FACTORING problem:

Instance: An integer n (written in decimal)
Solution: A list of primes whose product equals n

Describe a decision problem X that is polynomial-time equivalent to FACTORING. (Given a black box for X, you can efficiently solve FACTORING. Given a black box for FACTORING, you can efficiently solve X.)

Exercise 91. The GREEDY heuristic for VERTEX COVER repeats the following step until the graph has no more edges: "Select a vertex v of highest degree, put v into the vertex cover, and delete all edges incident to v." Ties are broken arbitrarily. Show that the approximation guarantee of this GREEDY heuristic is (a) strictly worse than 2; (b) strictly worse than 1000.

Exercise 92. Find TSP instances (with triangle inequality) that illustrate that our analysis of the Double-Tree algorithm (worst case ratio 2) and our analysis of the Christofides algorithm (worst case ratio 3/2) are essentially tight.

Exercise 93. Show that for the general TSP (without triangle inequality), the existence of a polynomial time approximation algorithm with worst case ratio 2 would imply P=NP.

Exercise 94. An instance of the METRIC STEINER TREE problem consists of a complete graph on a vertex set $R \cup S$, and non-negative weights $w(e)$ on the edges that satisfy the triangle inequality. The vertices in R are called *required* vertices, and the vertices in S are called *Steiner* vertices. The goal is to find a minimum weight tree that contains all required vertices and some of the Steiner vertices.

A simple approximation algorithm uses the minimum spanning tree for the vertices in R as approximation of the Steiner tree. Determine the worst case ratio of this algorithm.

Exercise 95. An instance of BIN PACKING consists of n items with real sizes $a_1, \ldots, a_n \in [0, 1]$. The goal is to pack these items into the smallest possible number of unit size bins. The FIRST FIT algorithm for BIN PACKING works through the item list $a_1, \ldots, a_n$ one by one, and always packs the current item into the earliest (leftmost) bin into which it will fit.
(a) Show that the worst case ratio of FIRST FIT is at most 2.
(b) Show that the worst case ratio of FIRST FIT is at least 5/3.

Exercise 96. Show that for the BIN PACKING problem, the existence of a polynomial time approximation algorithm with worst case ratio 4/3 would imply P=NP.

Exercise 97. An instance of the MAKESPAN minimization problem consists of n jobs with (positive integer) lengths $p_1, \ldots, p_n$ and of m identical machines. The goal is to assign the jobs to the machines, so that the largest work-load is minimized. Consider the following approximation algorithm for MAKESPAN that improves on LIST SCHEDULING: Sort and rename the jobs, such that $p_1 \geq p_2 \geq \cdots \geq p_n$, and then apply LIST SCHEDULING to the sorted instance. Analyze this improved approximation algorithm LSI for MAKESPAN with $m = 4$ machines.

(a) Show that LSI has worst case ratio at most 5/4.
(b) Find a bad instance that matches this ratio 5/4.

Exercise 98. An instance of the 3-SET PACKING problem consists of a ground set X together with a system $S_1, S_2, \ldots, S_m$ of 3-elements subsets of X. The goal is to find a maximum cardinality subsystem of pairwise disjoint subsets.

Find a polynomial time approximation algorithm that finds a solution whose cardinality is at least 1/3 of the optimal cardinality.

Exercise 99. Give a polynomial time approximation algorithm with worst case ratio 1/2 for the ACYCLIC SUBGRAPH problem: An instance consists of a directed graph $G = (V, A)$. The goal is to find a maximum cardinality subset of the arcs that induces an acyclic subgraph.

Exercise 100. Design a polynomial time algorithm that takes as input a 3-colorable graph G and finds a proper coloring of G with $O(\sqrt{n})$ colors. (Hence: the input graph is guaranteed to be 3-colorable, but the corresponding 3-coloring is not known.)

Exercise 101. An instance of the MAX CUT problem consists of an undirected graph $G = (V, E)$. The goal is to find a partition $V = L \cup R$ that maximizes the number of edges between L and R.

The FLIP algorithm starts with the trivial partition with $L = V$ and $R = \emptyset$. As long as the objective value can be improved by moving one vertex from L to R or from R to L, FLIP moves the corresponding vertex. When there is no further improvement possible, FLIP outputs the current partition. Show that FLIP has worst case ratio 1/2.

Exercise 102. An instance of the KNAPSACK problem consists of a knapsack of capacity W together with n items with positive integer weights $w_1, \ldots, w_n \leq W$ and positive integer profits $p_1, \ldots, p_n$, The goal is to find a subset $S \subseteq \{1, \ldots, n\}$ of the items with total weight at most W that maximizes the total profit.

(a) The GREEDY algorithm first sorts the items by decreasing profit to weight ratios, and then greedily selects and packs items in this order. Show that GREEDY has unbounded worst case ratio.

(b) MODIFIED GREEDY uses the same item ordering as GREEDY. It finds the lowest index k such that the weight of the first k items exceeds W, and outputs the better of $\{1, \ldots, k-1\}$ and $\{k\}$. Show that MODIFIED GREEDY has worst case ratio 2.

Exercise 103. State an ILP formulation of the KNAPSACK problem, and formulate the corresponding LP relaxation. Analyze the integrality gap of the LP relaxation.

Exercise 104. A graph $G = (V, E)$ with n vertices $v_1, \ldots, v_n$ forms an instance of the INDEPENDENT SET problem.

(a) Find an ILP formulation of INDEPENDENT SET: For every vertex v_k introduce a corresponding indicator variable x_k. Introduce appropriate constraints, and express the objective function in terms of the x_k.

(b) Relax your ILP formulation to an LP formulation with continuous variables x_k. Prove: For every integer R, there exists an instance of INDEPENDENT SET, for which the ratio between the objective values of the LP and the ILP is at least R.

(c) What can you say about the integrality gap of this LP relaxation?

Exercise 105. Consider the following ILP formulation of the MAKESPAN minimization problem on identical machines with job processing times $p_1, \dots, p_n$.

$$
\begin{array}{lll}
\min & C & \\
\text{s.t.} & \sum_{i=1}^{m} x_{ij} \;=\; 1 & \text{for } j = 1, \dots, n \\
& \sum_{j=1}^{n} x_{ij} p_j \;=\; L_i & \text{for } i = 1, \dots, m \\
& L_i \;\leq\; C & \text{for } i = 1, \dots, m \\
& p_j \;\leq\; C & \text{for } j = 1, \dots, n \\
& x_{ij} \in \{0,1\} & \text{for } i = 1, \dots, m \text{ and } j = 1, \dots, n
\end{array}
$$

Note that the variables x_{ij} are integral, while the variables L_i and C are continuous.

- **(a)** Show that this ILP correctly models the MAKESPAN minimization problem.
- **(b)** Formulate the corresponding LP relaxation.
- **(c)** Analyze the integrality gap in dependence of the number m of machines.

Exercise 106. An instance of the MAX BISECTION problem consists of an undirected graph with vertex set $V = \{1, 2, \dots, 2n\}$ and positive real edge weights $w(i,j)$ for $[i,j] \in E$. The goal is to partition V into two parts V_1 and V_2 of size n, so that the total weight of the edges between V_1 and V_2 is as large as possible. Consider the following ILP.

$$
\begin{array}{lll}
\max & \sum_{[i,j]\in E} w(i,j)\, z_{i,j} & \\
s.t. & z_{i,j} \;\leq\; x_i + x_j & \text{for } [i,j] \in E \\
& z_{i,j} \;\leq\; 2 - x_i - x_j & \text{for } [i,j] \in E \\
& \sum_{i=1}^{2n} x_i \;=\; n & \\
& x_i \in \{0,1\} & \text{for } i \in V \\
& z_{i,j} \in \{0,1\} & \text{for } [i,j] \in E
\end{array}
$$

- **(a)** Show that this ILP correctly models the MAX BISECTION problem.
- **(b)** Show that any feasible solution x and z of the ILP satisfies $z_{ij} = x_i + x_j - 2x_i x_j$.

Next, let us consider the LP relaxation of this ILP, where the integrality constraints $x_i \in \{0,1\}$ and $z_{i,j} \in \{0,1\}$ are relaxed to the continuous constraints $0 \leq x_i \leq 1$ and $0 \leq z_{i,j} \leq 1$. Furthermore we define the auxiliary value $F(x) := \sum_{[i,j]\in E} w(i,j)\,(x_i + x_j - 2x_i x_j)$.

- **(c)** Prove that any feasible solution x and z of the LP relaxation satisfies the inequality $F(x) \;\geq\; \frac{1}{2} \sum_{[i,j]\in E} w(i,j)\, z_{i,j}$.
- **(d)** Consider a solution for the LP in which the two variables x_i and x_j are both fractional. Argue that it is possible to increase one variable by $\varepsilon > 0$ and to decrease the other one by the same ε, such that the value of $F(x)$ does not decrease and one of the two variables becomes integer.
- **(e)** Use these arguments to design a polynomial time approximation algorithm for MAX BISECTION that yields at least $1/2$ of the optimal objective value.

Exercise 107. An instance of the MAX SATISFIABILITY problem consists of a set X of logical variables $x_1, \ldots, x_n$; a set C of clauses $c_1, \ldots, c_m$ over X; a positive real weight w_c for every clause $c \in C$. Throughout this exercise we will assume that C does not contain contradictory unit-clauses; this means that for any variable x, at most one of the two clauses (x) and $(\neg x)$ is in C. The goal is to find a truth-setting of the variables in X that maximizes the overall weight of all satisfied clauses. For example, for the three clauses $c_1 = (\neg x_1)$, $c_2 = (\neg x_2)$, $c_3 = (x_1 + x_2)$ with weights $w_1 = w_2 = 3$ and $w_3 = 4$, the optimal solution satisfies two clauses with an overall weight of 7.

This exercise analyzes the quality of the bound $W = \sum_{c \in C} w_c$ for the optimal objective value of MAX SATISFIABILITY without contradictory unit-clauses; clearly $\textsc{Opt} \le W$. Define $\phi = \frac{1}{2}(-1 + \sqrt{5}) \approx 0.618$ as the positive root of $\phi^2 + \phi = 1$.

- **(a)** Show that there always exist truth-settings with objective value at least ϕW. (Hint: Set x_i=TRUE with an appropriately chosen probability p_i. Show that the expected value of the overall weight of the satisfied clauses is at least ϕW.)
- **(b)** Use the method of conditional expectation with the result in (a), and get a deterministic polynomial time approximation algorithm with worst case guarantee ϕ.
- **(c)** Find sets C of clauses, for which $\textsc{Opt}$ comes arbitrarily close to ϕW.

Exercise 108. Consider n jobs J_j $(j = 1, \ldots, n)$ with positive integer processing times p_j on two identical machines. The goal is to find a schedule with machine loads L_1 and L_2 that minimizes the following objective value:

- **(a)** $|L_1 - L_2|$
- **(b)** $\max\{L_1, L_2\} / \min\{L_1, L_2\}$
- **(c)** $(L_1 - L_2)^2$
- **(d)** $L_1 + L_2 + L_1 \cdot L_2$
- **(e)** $\max\{L_1, L_2/2\}$

For each of these problems, you are asked to either design a PTAS or to find a convincing argument against the existence of a PTAS.