Resit Combinatorial Optimization (E_EORM_COPT), 11 December 2018.

- It is not allowed to use any books, notes, or calculator. Just pen and paper.
- **You may also obtain points for partial solutions.** Most importantly: describe the main arguments. Minor details and precise notation are less important. Sometimes, a picture can be a helpful support to the arguments.
- The table shows the maximum number of points per (sub-)question.

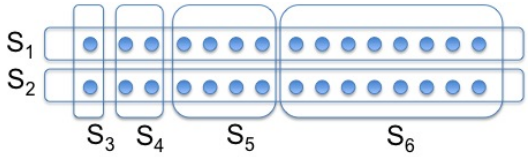
1a	1b	2	3	4	5a	5b	6a	6b	7a	7b	7c	8a	8b	Σ
1	2	6	4	6	3	3	3	3	3	2	1	4	4	45

---

1. This question is about the unweighted set cover problem.

(a) Describe the greedy algorithm for unweighted set cover. Now, suppose you apply it to the example below. What are the sets chosen and in what order? What is the ratio between the value of this solution and the optimal value?

(b) Argue that the approximation ratio of the greedy set cover algorithm is in general not better than $c \log n$ for some constant c, where n is the number of elements.
(You do not have to compute c. Just show that the ratio of greedy is $\Omega(\log n)$.)



2. A *tournament* $G = (V,A)$ is a complete, directed graph. That means, for every pair of vertices i, j, either arc $(i, j) \in A$ or arc $(j,i) \in A$ and not both. A *feedback vertex set* S for G is a subset of the vertices such that removing S and its adjacent arcs leaves an acyclic graph (a graph without directed cycles).

FEEDBACK VERTEX SET:

Instance:	Tournament $G = (V,A)$
Solution:	$S \subset V$ such that the graph induced by $V - S$ is acyclic
Value:	$\lvert S \rvert$
Goal:	Minimize $\lvert S \rvert$

Give a 3-approximation for finding a minimum feedback vertex set.

Hint: First show that if the remaining graph has no directed cycles of length 3, then it has no directed cycles at all. Then, formulate the problem as a set cover problem and use the fact that there is an f-approximation for set cover.

3. In the maximum *directed* cut problem we are given as input a directed graph $G = (V, A)$. Each arc $(i, j) \in A$ has nonnegative weight $w_{ij} \geq 0$. The goal is to partition V into two sets U and $W = V \setminus U$ so as to maximize the total weight of the arcs going from U to W (that is, arcs (i, j) with $i \in U$ and $j \in W$). Give a randomized $(1/4)$-approximation algorithm for this problem and give a proof for this ratio.

4. Consider the following algorithm for the scheduling problem $Pm||C_{\max}$ (the standard scheduling problem on m identical parallel machines.)

```
Step 1:  Find an optimal schedule for the longest 5m jobs.

Step 2:  Add the remaining jobs one-by-one as early as
possible and always assign to the least loaded machine.
```

Show that this is a 1.2 approximation algorithm for this scheduling problem. (You do not need to prove that it runs in polynomial time.)

5. (a) Show that the following problem is NP-hard:

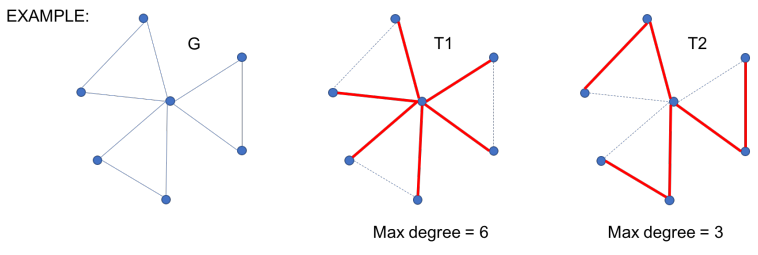
LONGEST PATH:

Instance:	Graph $G = (V, E)$
Solution:	A path containing each vertex at most once.
Value:	Length (number of edges) of the path.
Goal:	Maximize the length.

(b) Show that the following problem is NP-hard:

MINIMUM MAXIMUM DEGREE SPANNING TREE:

Instance:	Graph $G = (V, E)$
Solution:	A spanning tree T of G
Value:	The maximum degree in the spanning tree
Goal:	Minimize the maximum degree.



6. Section 4.2 of the book considers the scheduling problem: $1|r_j|\sum w_j C_j$ and presents a 3-approximation algorithm using LP-rounding.

(a) Describe this 3-approximation algorithm. That means, give the LP and explain how it is used to construct a schedule. (No proof needed, just describe the algorithm.)

(b) The LP of (a) has exponentially many constraints but it can be solved in polynomial time since it has a separation oracle. Describe this separation oracle.

7. This question is about the maximum satisfiability problem. An instance consists of m clauses C_j, each containing one or more literals.

(a) Describe a *randomized* $\frac{1}{2}$-approximation algorithm for the maximum satisfiability problem and give a proof for the approximation ratio.

(b) What will be the *expected* number of satisfied clauses when you apply the algorithm of (a) to the instance below? (It has 15 clauses and 4 variables in each clause.)

(c) Is there an assignment that satisfies all clauses of the instance below? *Hint: use (b).*

$$
\begin{array}{rrrrr}
C_1 = & x_1 \vee & \neg x_2 \vee & \neg x_3 \vee & x_4 \\
C_2 = & x_1 \vee & x_2 \vee & \neg x_3 \vee & x_4 \\
C_3 = & \neg x_1 \vee & x_2 \vee & x_3 \vee & x_4 \\
C_4 = & \neg x_1 \vee & x_2 \vee & \neg x_3 \vee & x_4 \\
C_5 = & x_1 \vee & \neg x_2 \vee & \neg x_3 \vee & \neg x_4 \\
C_6 = & \neg x_1 \vee & x_2 \vee & x_3 \vee & \neg x_4 \\
C_7 = & x_1 \vee & x_2 \vee & x_3 \vee & x_4 \\
C_8 = & \neg x_1 \vee & \neg x_2 \vee & \neg x_3 \vee & \neg x_4
\end{array}
\qquad
\begin{array}{rrrrr}
C_9 = & x_1 \vee & x_2 \vee & x_3 \vee & \neg x_4 \\
C_{10} = & x_1 \vee & x_2 \vee & \neg x_3 \vee & \neg x_4 \\
C_{11} = & \neg x_1 \vee & \neg x_2 \vee & \neg x_3 \vee & x_4 \\
C_{12} = & \neg x_1 \vee & x_2 \vee & \neg x_3 \vee & \neg x_4 \\
C_{13} = & x_1 \vee & \neg x_2 \vee & x_3 \vee & x_4 \\
C_{14} = & x_1 \vee & \neg x_2 \vee & x_3 \vee & \neg x_4 \\
C_{15} = & \neg x_1 \vee & \neg x_2 \vee & x_3 \vee & x_4
\end{array}
$$

8. (a) Chapter 6 of the book gives a Vector Program relaxation for the graph coloring problem. Give that vector program and show that for any 3-colorable graph, the value of the VP is at most -0.5.

(b) A k-cut of a graph $G = (V,E)$ is a partition of V into k sets, $V_1, \ldots, V_k$. The value of the k-cut is the number of edges that have endpoints in different sets of the partition. (The case $k = 2$ is simply called a cut of a graph.) Let G be a 3-colorable graph with 90 edges. Show how to use the VP to find a 4-cut that contains at least 80 edges (in expectation).

Appendix

Some problems that can be solved in polynomial time:

- Linear programs and Vector programs.
- Finding a minimum cut in a graph.
- Finding a maximum flow in a graph.
- Finding the shortest path between two points in a graph.
- Finding a minimum spanning tree in a graph.
- Finding a maximum matching in a graph.
- Finding a minimum cost perfect matching in a graph.

Some problems that are NP-hard/NP-complete:

- Vertex Cover
- Set Cover
- Hamiltonian Cycle
- Hamiltonian Path
- 3-Partition
- Knapsack
- TSP
- k-center / k-cluster
