Advanced Algorithms 14/15. Example exam + solutions.

- It is not allowed to use any books, notes, calculator ... just pen and paper.
- The table shows the maximum number of points per (sub-)question. The exam score is obtained by dividing the total number of points by 3.

1a	1b	1c	2a	2b	2c	3	4a	4b	5a	5b	5c	6a	6b	7a	7b	Σ
1	2	1	2	2	2	4	1	3	2	1	2	1	2	2	2	30

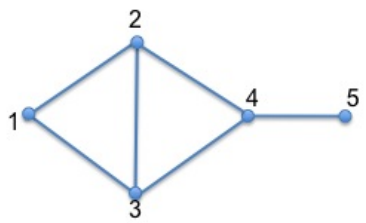
---

1. This question is about the vertex cover problem of chapter 1.

(a) Give an optimal vertex cover for the graph below.

(b) Write down the ILP for vertex cover for this example.

(c) For this example, give a solution to the LP-relaxation which has value strictly smaller than the value of the optimal vertex cover.



2. This question is about the unweighted set cover problem. An instance is given by a set of elements (items) $E = \{e_1, \ldots, e_n\}$ and subsets $S_1, \ldots, S_m \subseteq E$.

(a) Give the ILP for set cover and the dual of its LP-relaxation.

(b) Assume that each element appears in at most f sets (for some constant f). Now describe an f-approximation algorithm. (It may or may not use the dual).

(c) Prove that your algorithm is an f-approximation algorithm.

3. This question is about the parallel machine scheduling problem of chapter two. Here, we are given n jobs with processing time $p_1, \ldots, p_n$ and m identical machines. The problem is to assign the jobs to the machines such that the length of the schedule (= latest job completion time C_{max}) is minimized. Describe the list scheduling algorithm and prove that this is a 2-approximation algorithm.

4.

(a) Give the definition of a polynomial time approximation scheme (PTAS) for a maximization problem.

Consider the following maximization problem. All coëfficients v_j, a_j, b are non-negative integers and $a_j \leqslant b$.

$$\begin{aligned} \text{maximize} \quad & Z = \sum_{j=1}^{n} v_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_j x_j \leqslant b, \\ & x_j \in \{0,1\} \qquad j = 1, \ldots, n. \end{aligned}$$

Let ALG be some algorithm which solves this problem in $p(n) \cdot V$ time, where $V = \max_j v_j$ and $p(n)$ is some polynomial in n.

(b) You are given some constant $\varepsilon > 0$. Show how to use algorithm ALG to get a $(1-\varepsilon)$-approximation algorithm. How do you round numbers exactly? Show that your algorithm runs in polynimial time and give a short intuitive argument for why the value is at least $(1-\varepsilon)$ times optimal.

5. Consider the (unweighted) maximum satisfiability problem (Max Sat). For each clause C_j, denote by P_j the indices of the variables x_i that occur positively in C_j and let N_j be the indices of the variables x_i that are negated in the clause.

$$\begin{aligned} \max \quad & Z = \ldots\ldots \\ s.t. \quad & \ldots\ldots && \text{for all } j = 1 \ldots m, \\ & y_i \in \{0,1\} && \text{for all } i = 1 \ldots n, \\ & 0 \leqslant z_j \leqslant 1 && \text{for all } j = 1 \ldots m. \end{aligned}$$

(a) Complete the mixed ILP for Max Sat above.

Let $f(y)$ be a function on $[0,1]$ such that $1 - 4^{-y} \leqslant f(y) \leqslant 4^{y-1}$. Consider the following algorithm for the Max Sat problem.

Algorithm :
Step 1. Solve the LP-relaxation of the mixed ILP of question (a). $\rightarrow y^*, z^*, Z^*_{LP}$.

Step 2. Set each variable x_i to true with probability $f(y^*_i)$.

(b) Show that $\Pr(C_j \text{ is not satisfied}) \leqslant 4^{-z^*_j}$ for any clause C_j.

(c) Now use (b) to show that the algorithm is a $3/4$-approximation algorithm for Max Sat. You may use that the function $g(z) = 1 - 4^{-z}$ is a concave function on $[0,1]$. See the figure.

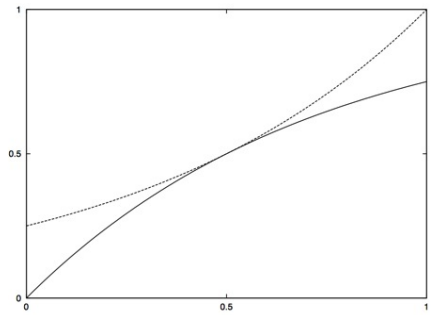


Figure 1: *The functions* $1-4^{-y}$ *and* 4^{y-1} *(for question 5).*

6. Consider the following minimization problem:

DEGREE BOUNDED SPANNING TREE:

Instance:	Graph $G=(V,E)$
Solution::	A spanning tree T of G
Value:	Maximum degree of T
Goal:	Find a solution with minimum value.

(a) It is well-known that the Hamiltonian path problem is *NP*-complete. Use this to show that the degree bounded spanning tree problem is *NP*-hard (*NP*-complete).

(b) Let $\alpha < 3/2$. Show that there is no polynomial time α-approximation algorithm for this problem , unless $P \neq NP$.

7.

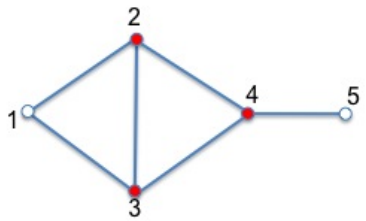
(a) Give a strict quadratic program for the maximum weighted cut problem and give its vector program relaxation.

(b) Give (draw) an unweighted graph for which the maximum cut has value 4 and for which the optimal value to the vector program relaxation has value strictly larger than 4. Draw the cut and give a solution for the VP relaxation with value larger than 4.

Solutions

(Some comments to answers in blue.)

1. (a) OPT $= 3$. For example: $\{2,3,4\}$. (More solutions possible.)



(b)

$$\begin{aligned}
\min \quad & x_1+x_2+x_3+x_4+x_5 \\
s.t. \quad & x_1+x_2 \geqslant 1 \\
& x_1+x_3 \geqslant 1 \\
& x_2+x_3 \geqslant 1 \\
& x_2+x_4 \geqslant 1 \\
& x_3+x_4 \geqslant 1 \\
& x_4+x_5 \geqslant 1 \\
& x_1,x_2,x_3,x_4,x_5 \in \{0,1\}.
\end{aligned}$$

(c) For example, $x_i = 0.5$ for all i. The LP-value is $2.5 < 3$.
(More solutions possible, such as, $x_1 = x_2 = x_3 = 0.5, x_4 = 1, x_5 = 0$.

2. (a)

$$\begin{aligned}
\text{(ILP)} \ \min \quad & Z = \sum_{j=1}^{m} x_j \\
s.t. \quad & \sum_{j:e_i \in S_j} x_j \geqslant 1 \qquad \text{for all } i = 1,\ldots,n \\
& x_j \in \{0,1\} \qquad \text{for all } j = 1,\ldots,m.
\end{aligned}$$

$$\begin{aligned}
\text{(D)} \ \max \quad & Z = \sum_{j=1}^{n} y_i \\
s.t. \quad & \sum_{i:e_i \in S_j} y_i \leqslant 1 \qquad \text{for all } j = 1,\ldots,m \\
& y_i \geqslant 0 \qquad \text{for all } i = 1,\ldots,n.
\end{aligned}$$

(b) We have seen several f-approximation algorithms. The easiest is LP-rounding:

Algorithm :
Step 1: Solve the LP-relaxation. $\rightarrow x_1^*, x_2^*, \ldots, x_m^*, Z_{LP}^*$
Step 2: Add j to the solution if $x_j^* \geqslant 1/f$.

(c) Clearly, the running time is polynomial since LP's can be solved in polynomial time.
Any solution is feasible since each item appears in at most f sets. That means, there are at most f variables in the constraint for e_i and at least one of the variables must has value $\geqslant 1/f$.

To show the factor f denote the rounded solution by $\hat{x}$, that means, $\hat{x}_j = 1$ if $x_j^* \geqslant 1/f$ and $\hat{x}_j = 0$ otherwise. In either case, $\hat{x}_j \leqslant f x_j^*$. The value of the solution found is

$$\sum_{j=1}^{m} \hat{x}_j \leqslant f \sum_{j=1}^{m} x_j^* = f Z_{LP}^* \leqslant f Z_{ILP}^* = f \textsc{Opt}.$$

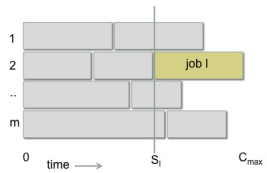
3. (Make a picture. Important is that you give the two bounds on OPT and explain how you use them. Notation is not so important.)

List scheduling:
Place the jobs in arbitrary order. Following this order, place the jobs one by one on the machines, always scheduling the jobs as early as possible.

Clearly, the algorithm runs in polynomial time and any solution is feasible. For the proof of the ratio we use the lower bounds

$$\textsc{Opt} \geqslant \max_j p_j \quad \text{and} \quad \textsc{Opt} \geqslant \frac{1}{m} \sum_j p_j. \tag{1}$$

Let l be the job that completes last in the schedule and let S_l be its start time.



Since all machines must be busy until time S_l: $S_l \leq \frac{1}{m} \sum_j p_j$. The length of the schedule is

$$S_l + p_l \leq \frac{1}{m} \sum_j p_j + p_l \leq \textsc{Opt} + \textsc{Opt}.$$

4. **(a)** PTAS: For each constant $\varepsilon > 0$ there is algorithm A_ε such that A_ε is a $(1-\varepsilon)$-approximation algorithm.

(b) You should notice here that this is exactly the knapsack problem.

Algorithm:
Round the instance. Then, solve the rounded instance by ALG and take that as the solution.

Rounding is the same as in the knapsack problem: $v_i' = \lfloor v_i/\mu \rfloor$ with $\mu = \frac{\varepsilon V}{n}$.

Any solution is feasible since only the v_i's are rounded.

The maximum value in the rounded instance is $V' = \lfloor V/\mu \rfloor = \lfloor n/\varepsilon \rfloor$. The running time for algorithm ALG is $p(n)V'$ which is polynomial for constant ε.

The error made by rounding is at most μ for each j. The total error is at most $n\mu = \varepsilon V$. Next, we argue that $V \leq \text{OPT}$. Let $v_k = V$. It is given that $a_j \leq b$ for all j. Therefore, a feasible solution is $x_k = 1$ and $x_j = 0$ for all $j \neq k$. That implies $\text{OPT} \geqslant v_k = V$.

5. (a)

$$\begin{aligned}
\max \quad & Z = \sum_{j=1}^{m} z_j \\
s.t. \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geqslant z_j && \text{for all } j = 1 \ldots m, \\
& y_i \in \{0,1\} && \text{for all } i = 1 \ldots n, \\
& 0 \leqslant z_j \leqslant 1 && \text{for all } j = 1 \ldots m.
\end{aligned}$$

(b)

$$\begin{aligned}
\Pr(C_j \text{ is not sat.}) &= \prod_{i \in P_j} (1 - f(y_i^*)) \prod_{i \in N_j} f(y_i^*) \\
&\leqslant \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^* - 1} \\
&= 4^{-(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} 1 - y_i^*)} \\
&\leqslant 4^{-z_j^*}
\end{aligned}$$

(c) From (b):

$$\Pr(C_j \text{ is sat.}) \geqslant 1 - 4^{-z_j^*}.$$

It is given that $g(z) = 1 - 4^{-z}$ is concave on $[0,1]$. We have $g(0) = 1 - 4^{-0} = 0$ and $g(1) = 1 - 4^{-1} = 3/4$. (See figure) Thus, $1 - 4^{-z} \geqslant \frac{3}{4} z$ on $[0,1]$.

$$\Pr(C_j \text{ is sat.}) \geqslant \frac{3}{4} z_j^*.$$

The expected number of satisfied clauses is

$$\sum_{j=1}^{m} \Pr(C_j \text{ is sat.}) \geqslant \sum_{j=1}^{m} \frac{3}{4} z_j^* = \frac{3}{4} Z_{LP}^* \geqslant \frac{3}{4} \text{OPT}.$$

6. (a) We show that the Hamiltonian path problem (HP) can be reduced to the degree bounded spanning tree problem. Assume we have an algorithm that solves the degree bounded spanning tree problem. Then we can solve HP by applying this algorithm. Let G be a graph and let OPT_G be the optimal value for the degree bounded spanning tree problem for G. Then the following equivalence holds,

$$G \text{ has a Hamiltonian path} \Leftrightarrow \text{OPT}_G = 2.$$

(b) Assume there is such an algorithm. Let G be a graph and let ALG_G be the value returned by the algorithm. Then the following relations hold:

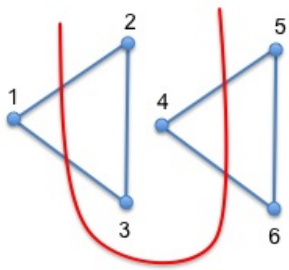
$$\begin{aligned}
&G \text{ has a Hamiltonian path} \Rightarrow \text{OPT}_G = 2 \Rightarrow \text{ALG}_G \leq \alpha \text{OPT}_G < 3. \\
&G \text{ has no Hamiltonian path} \Rightarrow \text{OPT}_G \geqslant 3 \Rightarrow \text{ALG}_G \geqslant 3.
\end{aligned}$$

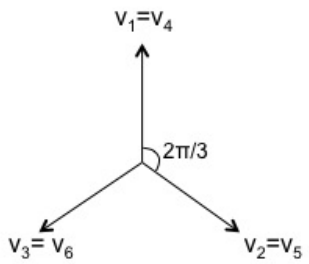
We conclude the algorithm can be used to solve HP in polynomial time. This is not possible, assuming $P \neq NP$.

7. (a)

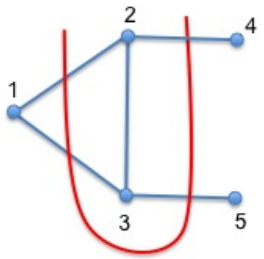
$$\begin{aligned}
\text{(QP)} \quad \max \quad & \tfrac{1}{2}\sum_{(i,j)}(1-y_iy_j)w_{ij} \\
s.t. \quad & y_i^2 = 1 \qquad\qquad i=1,\dots,n. \\
\text{(VP)} \quad \max \quad & \tfrac{1}{2}\sum_{(i,j)}(1-v_i\cdot v_j)w_{ij} \\
s.t. \quad & v_i\cdot v_i = 1,\ v_i\in\mathbb{R}^n \qquad i=1,\dots,n.
\end{aligned}$$

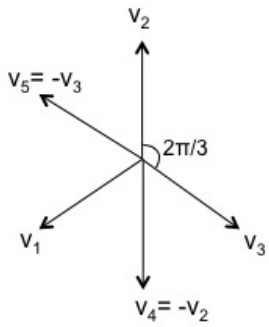
(b) An easy example is to take two triangles. The value of the VP solution in the picture is $\frac{1}{2}6(1--0.5)=4.5$





That graph above is not connected but that is OK. Here is also a possible example with a connected graph: The VP solution in the picture has value $\frac{1}{2}(3(1--0.5)+2(1--1))=4.25$.





Here, the vector solutions are drawn in the plane while the vectors are actually in $\mathbb{R}^n$ (in the examples $n=6$ and $n=5$). The given solutions have the property that all vectors lie in a two dimensional subspace and are therefore easy to draw. VP solutions with larger value may be obtained if we use the full dimension. However, that is not so easy to draw in a picture.