Lehrstuhl für Informatik I
– Algorithmen und Komplexität –
Universität Würzburg

Würzburg, 9 November 2022

Johannes Zink
Oksana Firman

# Exercise Sheet #4

# Advanced Algorithms (WS 2022/23)

**Exercise 1 – Randomized Max Cut**

Let G be the graph shown in Figure 1 (a). Apply the following steps of the algorithm RANDOMIZEDMAXCUT from the lecture.

a) Formulate the quadratic program QP, whose optimal solution gives a maximal cut for G; i.e. give the variables, the constraints and the objective function with the respective values. **4 Points**

b) Formulate its relaxation $QP^k$, for $k = 2$. **1 Point**

An optimal solution for $QP^2$ is shown in Figure 1 (b). For the vectors $x^1, x^2, \dots, x^6$ we have $x^1 = x^3 = (-1, 0)$, $x^2 = (1, 0)$, $x^4 = (0, 1)$, $x^5 = (0, -1)$, and $x^6 = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$.

c) List all cuts that RANDOMIZEDMAXCUT could compute from this solution and calculate their weight. What is the expected value compared to the optimal solution? **3 Points**

d) We use the randomly chosen vector r to get from a solution of $QP^2$ to a cut in G. Can we instead just pick r efficiently such that we get the best cut? **1 Point**

e) Why do we pick the vector r at random for $QP^n$ instead of taking the one maximizing the cut? **1 Point**

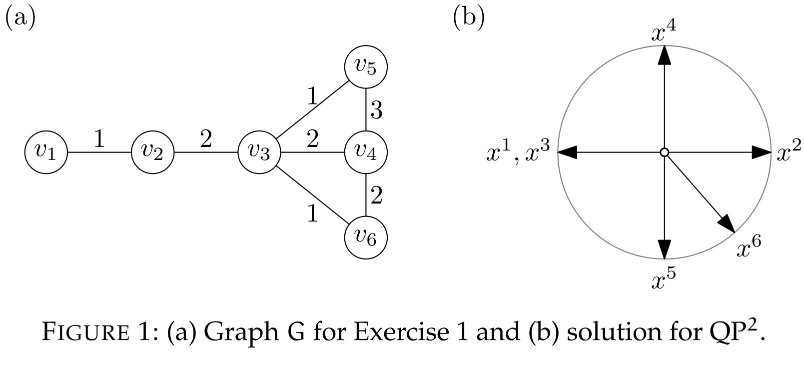


FIGURE 1: (a) Graph G for Exercise 1 and (b) solution for $QP^2$.

**Exercise 2 – QP for MAX-2SAT**

Given a conjunctive normal form formula $f$ of Boolean variables $x_1, \ldots, x_n$ and non-negative weights $w_c$ for each clause $c$ of $f$, the MAX-SAT problem asks for a truth assignment to the variables such that the total weight of satisfied clauses is maximized. For the problem MAX-2SAT the clauses $c_1, \ldots, c_m$ are restricted to contain at most 2 literals, e.g. $(x_1 \vee \neg x_3)$. Not just MAX-SAT, but even MAX-2SAT is NP-hard.

Give a quadratic program for MAX-2SAT. **5 Points**

**Exercise 3 – Deterministic 0.5-approximation for MaxCut**

In the lecture we saw a randomized 0.5-approximation algorithm for the unweighted MAXCUT problem. We now want to derandomize this algorithm with the method of conditional probabilities described next.

Consider the first vertex $v_1$ for which we flipped a coin. We now want to decide deterministically, whether we should put $v_1$ in $S$ or not. For this, we consider the expected weight $E[W]$ of the cut where $v$ is set to either be in $S$ or not in $S$ but the vertices $v_2, \ldots, v_n$ are still assigned randomly. More precisely, we put $v$ in $S$ if and only if $E[W|v_1 \in S] \geq E[W|v_1 \notin S]$. Note that $E[W] = (E[W|v_1 \in S] + E[W|v_1 \notin S])/2$. Hence, by our choice $A_1 \in \{S, V \setminus S\}$, we know that $E[W|v_1 \in A_1] \geq E[W] \geq 0.5\text{OPT}$. We can repeat this process with $v_2$ and put it in $A_2 \in \{S, V \setminus S\}$ based on whether $E[W|v_1 \in A_1, v_2 \in S] \geq E[W|v_1 \in A_1, v_2 \notin S]$. In fact, we can repeat this for all the reamining vertices $v_3, \ldots, v_n$. However, to develope an algorithm, we need to be able to efficiently compute $E[W|v_1 \in A_1, \ldots, v_i \in A_i]$.

Describe how we can compute $E[W|v_1 \in A_1, \ldots, v_i \in A_i]$ efficiently. Derive a simple algorithm from this. **5 Points**

---

Please hand in your solutions on Wuecampus until the beginning of the next lecture, that is 14:15 on Wednesday, November 16.