Universität Bremen

FB3: Mathematik/Informatik

Dr. Martin Böhm
Dr. Ruben Hoeksma
Prof. Dr. Nicole Megow

Summer 2019

Approximation Algorithms

Homework Sheet 4 (Deadline 23. 05. 2019 12:00 (paper) or 23:59 (email).)

All exercises must be done individually. Feel free (and encouraged) to discuss among each otherut each solution must be written independently. Two or more submissions found with exactly the same solution on any of the exercises will be awarded no points for the entire exercise sheet. The same holds in case of directly copying from any other source.

Exercise 1 (*10 points*). Consider the problem MaxSat, which we state as follows:

Input: A list of n variables and a list of m clauses on those variables, e.g.:

$$\{1, 2, 3, 4, 5\}, \{(x_1 \vee \neg x_5), (x_2 \vee \neg x_4), (x_4)\}.$$

All clauses have only `or` as their binary logical operation.

Output: Any assignment, possibly evaluating some clauses to `False`.

Goal: Maximize the number of the clauses that are satisfied.

We will analyze the following algorithm:
"Try setting all variables to 0 and compute how many clauses we have satisfied. Then, try setting all variables to 1 and compute how many clauses we have satisfied. Take the bigger of the two values and return it as the approximation."

a) This algorithm is clearly a 1/2-approximation; your task is to create an instance (or a set of instances, one for every r') which shows that this algorithm is not an r'-approximation for any $r' > 1/2$.

b) Let us consider any *constant-testing* algorithm – such an algorithm does the same thing as the one described above, but instead of two assignments, it tests c pre-defined assignments, where c is a constant not dependent on the input. (An assignment is any infinite sequence of 0/1 values, where we assign the first value to x_1, the second value to x_2 and so on, until we run out of variables. In other words, the first algorithm tests assignments $0, 0, 0, \ldots$ and $1, 1, 1, \ldots$.)

What is the tight approximation ratio of any constant-testing algorithm? Again, you need to find a number r_2 such that some constant-testing algorithm is an r_2-approximation and prove by creating an instance (or a set of them) that *no* constant-testing algorithm is strictly better than an r_2-approximation.

Exercise 2 (*10 points*). At the exercise session we have formulated the LP formulation for Max Dicut, which is the following problem:

On the input we get a directed graph $G = (V, \vec{E})$ and a non-negative weight function on the edges. Our task is to find a subset of vertices S so that $\vec{E}(S, V \setminus S)$ (the edges directed from S to the rest) have maximum possible weight.

Our LP relaxation was:

$$\begin{aligned} \max \quad & \sum_{ab \in \vec{E}(G)} w_{ab} z_{ab} \\ \text{s.t.} \quad & \forall ab \in \vec{E}(G) \colon z_{ab} \leq x_a \\ & \forall ab \in \vec{E}(G) \colon z_{ab} \leq (1 - x_b) \\ & \forall v \in V(G) \colon x_v \geq 0, x_v \leq 1 \\ & \forall ab \in \vec{E}(G) \colon z_{ab} \geq 0, z_{ab} \leq 1 \end{aligned}$$

Prove that the following algorithm is a 1/2-approximation:

a) Compute the optimal solution vectors z^*, x^* to the LP above.

b) Choose each vertex v_i into set S independently with probability $1/4 + x_i^*/2$.

c) Return the randomly generated cut S and its boundary edges.

Hint: Your starting point should be:

$$Pr[\text{edge } ab \text{ is on the boundary}] = Pr[a \in S \wedge b \notin S] = \ldots$$

and your final inequality should be:

$$\ldots \geq \frac{1}{2} z^*_{ab}.$$

You also need to argue why proving this inequality proves the whole 1/2-approximation.

For additional hints, you can also read the proof in *Section 5.4: Randomized rounding* in the book Williamson, Shmoys; even though it is for Max Sat, the approach is the same.