Universität Bremen

FB3: Mathematik/Informatik

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

Summer 2019

Approximation Algorithms

Exercise Sheet 4 (16. 05. 2019)

Exercise 1. **Card tricks.** We have 52 cards, half red, half black, randomly shuffled (a uniformly random permutation). We now reveal one card after another. You (as the algorithm) have two options: either say "I want the top card" – you win if it is red, you lose if it is black, and the game ends after one guess – or say "Keep showing me cards" – and then we reveal the next top card in the deck, allowing you to guess on the next card.

We are interested in the best algorithm, that is one maximizing the probability that it wins.

a) What is the probability of winning for the algorithm $Fil \equiv$ "always guess on the first card"? And what is the probability for $La \equiv$ "always guess on the last card"?

b) Show that the probability of the next card being red is not always the same value – that the sequence of already revealed cards actually changes the probability of the next card being red.

c) Consider the following algorithm that aims to be better than La: *"We keep revealing cards until we have more black cards revealed than red cards revealed. In such a situation, we stop: clearly in the deck there remain more red cards and our probability of winning on the next card is strictly more than* $1/2$*. This situation is very likely to happen; after all, the expected number of red cards and black cards in the revealed section is* 0 *on even turns, so the color balance has to fluctuate around that. In the worst case when this never happens, we pick the last card, so we are at least as good as* La*."*

   Try to brainstorm a few informal/intuitive arguments that show or disprove that this new algorithm is better than La.

d) Our main and final task: find an algorithm which chooses the red card with probability strictly more than $1/2$ – or prove that no such algorithm exists.

Exercise 2. We now consider Max Dicut. 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.

Suggest a probabilistic $\frac{1}{4}$-approximation algorithm for Max Dicut. This should be fairly easy, and it will not require linear programming.

Exercise 3. Let us try to improve on our algorithm for Max Dicut:

a) Suggest a natural $\{0,1\}$-integer program solving Max Dicut.
Hint: Use two kinds of variables similarly to Max Sat.

b) Choose each vertex v_i with probability $1/4 + x_i^*/2$, where x_i^* is the optimum of the linear relaxation of the previous integer program. Show that it is a 1/2-approximation.