INTRO TO APPROXIMATION, CLASS 3

probability in the hands of computer science

EXERCISE ONE **Max-k-Cut.** On the input for MAXIMUM k-CUT we get an undirected graph G and weights on the edges w: $E(G) \to \mathbb{R}^+$. Our goal is to partition the vertices into k disjoint sets $V_1, V_2, \ldots, V_k$ so that we minimize the sum of all "multicolored" edges (those that go from any one set to any other set).

Suggest and analyze a $\frac{k-1}{k}$-approximation algorithm for MAXIMUM k-CUT.

EXERCISE TWO **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.

1. What is the probability of winning for the algorithm $Fi \equiv$ "always guess on the first card"? And what is the probability for $La \equiv$ "always guess on the last card"?
2. 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.
3. Using the law of total probability, express E[algorithm wins] = P[algorithm wins] for some algorithm A.
4. 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.
5. 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 THREE **Maximizing satisfiability.** In the maximization problem MAX-SAT we get on input a Boolean formula (expression) in the CNF form $((x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge \ldots)$. Our task is not to check whether the entire clause is satisfiable, but to maximize the number of clauses that are satisfied.

1. What if we try a uniformly random assigment – setting each variable to 1 with probability 1. Compute the expected number of satisfied clauses in this case, and give an instance where the approximation ratio is tight. Does the ratio improve if we only allow inputs with at least two (or three) variables in every clause?
2. Is MAX-k-SAT as hard as k-SAT? No, it is not. First let us verify that 2-SAT (on input we have a CNF formula with at most two literals per clause, and we have to check whether the formula is satisfiable) is polynomial-time solvable.
3. Show that MAX-2-SAT (On input we get a number c and a CNF formula with at most two literals per clause and we have to decide whether there is an assignment satisfying at least c clauses) is NP-hard.
   Hint: There is a direct reduction from 3-SAT to MAX-2-SAT. Consider a clause with say three

literals and create several (about 10) new 2-SAT clauses. The idea here is as follows: if the algorithm can satisfy c of the new clauses, then we can create an assignment satisfying the original clause, too – and any unsatisfying assignment can only satisfy $c-1$ of the new clauses. *Second hint:* Some more variables are needed to build 10 clauses. Create an artificial variable d with the meaning „the clause is satisfied“. The hard part is to encode this into clauses of size two ...