INTRO TO APPROXIMATION – HW2 greedy algs, LP rounding, and more

Every task is worth two points. Deadline: **24. 12. 2016 08:00** via email. Please send PDFs only. High-quality scans (with high-quality handwriting) are acceptable, *but only in the PDF format.* Do not be afraid to email me or ask me if any task is unclear to you.

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

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

EXERCISE TWO We modify SCHEDULING as follows: there are m machines as before, but k of them process jobs with twice the speed, and $m-k$ with original speed (speed one). Our goal is the same: to schedule all the jobs so that the makespan (the time when the last machine finishes its work) is minimized.

For this problem, we can apply the same greedy algorithm that we used for SCHEDULING at the lecture: take jobs one by one and schedule each one on the machine where it finishes first.

Fully analyze this algorithm in the two speed setting. By that we mean prove that the algorithm is always a c-approximation and create an instance (or a sequence of instances) where the algorithm is no better than a c-approximation. Keep in mind that the optimum solution has k fast machines as well!

In this task you do not need to deal with asymptotics, a limit answer is sufficient (for instance, it is enough to show that the algorithm could be a 2-approximation, there is no need to compute some exact bound like $2 - \frac{1}{k}$).

EXERCISE THREE Consider the problem of GRAPH BALANCING, where you get an undirected graph G with weights on the edges $p\colon E(G) \to \mathbb{R}^+$. Our goal is to make the graph directed so that the most loaded vertex (the vertex with the most weight directed towards it) has minimum possible load. Formally we seek an orientation of the edges which minimizes the goal function $u = \max_{v\in V} \sum_{e\in E; e \text{ directed to } v} p(e)$.

Suggest and analyze a 2-approximation algorithm for the problem. (Tip: Linear programming might come in handy.)

EXERCISE FOUR Consider graphs with the maximum degree limited by a constant, i.e. $\Delta(G) \leq \Delta$. A greedy algorithm can color this graph easily with $\Delta + 1$ colors – but it is not easy to see how to parallelize it.

Suggest a fast randomized parallel algorithm, which can properly color the graph with $\Delta + 1$ colors with very high probability.

Think of Δ as a parameter of your algorithm that fits into an integer – your parallel computers know Δ in advance and can certainly compute things based on Δ in short time, but Δ^{Δ} messages passed between computers should not be considered as "$O(1)$ communications."

Tip: You may want to build on the material from the lectures.

BONUS EXERCISE Consider again the problem of GRAPH BALANCING. Prove that no algorithm can reach an approximation ratio lower than $3/2$ unless $P = NP$.

Tip: There may be many ways to prove this bound; here is a hint for a possible way: you can try to model one famous NP-complete problem using only a graph with edge weights 1 and $1/2$. Then, try to argue that if an approximation algorithm gets a solution with approximation factor better than $3/2$, then it must solve the NP-complete problem itself.