CSC2420: Algorithm Design, Analysis and Theory **Fall 2017**

Lecture 8: New Material

Lecturer: Nisarg Shah *Scribe: Lily Li*

8.1 Online Bipartite Matching

We have a fixed set V and an online set U. At each step, some u_i arrives and gets matched to an element of V, if possible. The naive approach is to pick a random vertex to match for u_i. Notice that the resulting matching is maximal (and an is a 1/2-approximation).

A better algorithm is known as the ranking algorithm: first the v_i's are given a ranking. As the u_j's come in, match u_j to its highest ranked neighbor v_i. This algorithm provides a $1 - 1/\epsilon$ approximation.

Lets go into the proof: First start with an optimal perfect matching m^*. Then for a random ordering σ of V let m_σ be the matching produced by the ranking algorithm. Note two claims: (1) if v of priority t is unmatched, then u (which v gets matched to in m^*) must get matched to some higher priority vertex under σ and (2)... this is a bit more annoying.

Here is another proof using LP relaxation and duality. The algorithm (Devanur et al) starts with the solution by the ranking algorithm. Then show that the dual solution is better than the optimal solution which is better than the primal. Since the ranking algorithm depends on the randomization σ the value is an expectation. Again we have two claims: (too slow!)

There is actually some setup necessary so you need to get that if you wanted the exact proof.

8.1.1 Better Bounds

We know that for the adversary from above gives a lower bound of $1 - \frac{1}{\epsilon}$. Thus we need to weaken the adversary to improve the bound. Suppose the adversary decides the graph but the distribution of the u_i's i.i.d (with possibly a known non-uniform distribution). This turns out that the ROM model is approximately the same with unknown distribution which is worse than the know distribution model.

8.2 Randomization to reduce Running Time

8.2.1 Exact 2-SAT

First the deterministic algorithm. This is simple: first satisfy all your unit clauses. What we end up with is a set of two term clauses. From each clause we derive an edge in a directed graph (nodes are all x and $\bar{x}$ for all variables x). There is an assignment if there is no path from x to $\bar{x}$ or vice versa. Running time $O(n^3)$.

Next the random walk 2-SAT: (we have seen this result before!) First set every clause to false, then for each clause pick one term and flip it. Observe that the number of satisfied variable changes by one at each step (one closer to the actual assignment τ or one farther to τ). Notice the probability of improving (getting closer to τ) is no worse than $\frac{1}{2}$. So, imagine a drunkard's walk from some $i \in \{0, ..., n\}$ (think of this as the hamming distance from your current assignment from τ) down to 0. The expected running time is $O(n^2)$.