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

Lecture 2: Priority Algorithms

Lecturer: Allan Borodin, Nisarg Shah *Scribe: Lily Li*

2.1 Continuing on from Last Time...

Also called myopic algorithms. This is a continuation of the discussion from last week. Recall the make-span problem from before. You need to impose a time limit — say polynomial — on determining the order of the objects. Otherwise you can just see all the objects.

Definition 2.1 *Consider a deterministic fixed order priority algorithm. We use this framework to argue negative results thus we view the model as a game between the algorithm and an adversary (oblivious not adaptive). Initial there is some (possibly infinite) set $\mathcal{J}$ of potential inputs. The algorithm choses a total ordering π on $\mathcal{J}$. The adversary chooses $\mathcal{I} \subset \mathcal{J}$ to be the input of the priority algorithm. The input items are then ordered according to π. Note: the ordering is local. Finally the algorithm executes on the input seeing the items in order.*

In the make-span case you can think of π as ordering the jobs in decreasing order. The adaptive case just has π determined by the adversary.

Notice here that we do not force each of these steps to have some time complexity. Further we do not consider advice. As it turns out, that really helps even though you might only get one bit of advice.

Here is an example of a problem that can be solved by this framework: interval scheduling problem (ISP). It turns out that weighted interval problem cannot be solved with any constant approximation using greedy algorithm, *but* it can be solved optimally by some other method.

Example 2.2 *Do you remember what it greedy algorithm for this problem? Answer: greedy by earliest finishing time. Notice here that for this problem tie-breakers do not matter. For proportional values (weight is length) there is a 3-approximation greedy algorithm (there is some intuition for why it should be 3, can you see it?). Strangely if you add a machine so you have to choose the machine then schedule, there is a greedy 2-approximation.*

Multipass Algorithm: um...

2.2 Greedy Algorithms for Set Packing

Here we consider combinatorial auctions. Given n subsets $S_1, ..., S_n$ from a universe U of size m. The goal is to find disjoint sub collections of S so as to maximize $\sum_{S_i \in S} w_i$. In the s-set packing problem we have $|S_i| \leq s$ for all i.

There are two natural greedy algorithms what are they? Unfortunately both of these greedy algorithms only produce a s-approximation.

2.3 Greedy Algorithm for Vertex Cover

Here we want to consider cases where the edges are weighted. If the edges are weighted then there are many 2-approximates. There is a conjecture that there are *no* $2 - \epsilon$ approximation. For the weighted case Clarkson's greedy algorithm is the one which gives you the 2-approximation.

2.4 Priority Stack Algorithm

Here the algorithm will maintain a stack of items (possibly with conflicts) that we might want to keep. At the end the algorithm pops the stack and resolves the conflicts. Apparently, with the appropriate rules for keeping stuff on stack, we can get a pretty good approximation.