CIS 320: Algorithms September 8, 2021

Lecture 2

Lecturer: Aaron Roth *Scribe: Aaron Roth*

The Assignment Problem

In this class we will be concerned with computationally efficient *problem solving*. One of the most important ideas in theoretical computer science—obvious once you get used it it, but surprisingly underappreciated in many other fields—is the distinction between the *class of problems* and the *solution* to those problems. A class of problems—and what constitutes a solution—should be defined independently of the method of solving it. A single problem may have multiple methods of solution—i.e. *algorithms*—that we can evaluate along different axes, such as running time, the degree of approximation of the solution, and others. In this lecture we'll give a simple example of class of problems, and an intuitive algorithm for solving them, that will get us used to algorithm analysis. Later in the class we will learn another method for solving the same problem that we will be able to compare with in retrospect.

Here is the abstract class of problems we will study:

Definition 1 *An instance of a bipartite matching problem is given by a set of* n left hand *vertices* L*, a set of* m right hand *vertices* R*, and a weight function* $w : L \times R \to \mathbb{R}_{\geq 0}$*, specifying a weight* $w(u, v)$ *for each pair of vertices* $u \in L$ *and* $v \in R$*.*

Think of $w(u, v)$ as representing the value or cost that will result from pairing u and v. For example, we might imagine that L corresponds to a set of students, and R corresponds to a set of dorm rooms, where $w(u, v)$ represents how happy student u will be if assigned to room v. We will wish to assign students to rooms to *maximize* value.

Implicit in this discussion is that we will somehow be *matching* or *assigning* elements of L to elements of R. This is formalized below:

Definition 2 *A* matching *is a mapping* $\mu : L \to R \cup \emptyset$ *such that each vertex in* L *is matched to at most one vertex in* R*, and no vertex in* R *is matched to more than one vertex in* L*. In other words, for every* $v \in R$*, and for every* $u, u' \in L$*,* $\mu(u) = v \Rightarrow \mu(u') \neq v$*.*

Here, $\mu(u) = v$ *"means" that* u *is matched to* v*, and* $\mu(u) = \emptyset$ *"means" that* u *is unmatched.*

A matching μ is a feasible solution to a bipartite matching problem. We can evaluate the quality of a solution by the weight of a matching, which we want to maximize. Here, we will interpret unmatched vertices in L as contributing nothing to the weight of the matching: $w(u, \emptyset) = 0$ for all $u \in L$.

Definition 3 *Given an instance* (L, R, w) *of a bipartite matching problem and a matching* μ*, the weight of the matching is given as:*

$$w(\mu) = \sum_{u \in L} w(u, \mu(u))$$

The weight of the optimal *matching is defined as:*

$$OPT = \max_{\mu} w(\mu)$$

where the maximum is taken over the set of all matchings.

An algorithm for the bipartite matching problem will take as input an instance (L, R, w) of the bipartite matching problem and output a matching μ. We will evaluate algorithms based on things like their running time and on the *quality of the solution* that they output. Algorithms that always output matchings μ such that $w(\mu) = \text{OPT}$ will be called exact algorithms for the bipartite matching problem, but algorithms that can guarantee *approximately* optimal solutions can also be interesting if e.g. they have running time advantages.

Finally, we introduce an important special case of the bipartite matching problem that corresponds to binary weights:

Definition 4 *An instance (L, R, w) of the bipartite matching problem is* unweighted *if for all $u \in L, v \in R$, $w(u, v) \in \{0, 1\}$.*

An unweighted matching problem corresponds to a setting in which certain matches are either feasible or not. Here OPT simply counts the maximum number of simultaneous feasible matches, which we want to maximize.

We'll now see an algorithm for finding approximately optimal bipartite matchings, that we can also use to compute exactly optimal matchings in unweighted matching problems. The algorithm itself will use a fiction of "prices", and will be very intuitive — but of course the problem itself — and hence the solution do not in fact make any reference to money. But to think about the algorithm, imagine that we are auctioning off the dorm rooms (vertices in R) to the students (vertices in L). Initially, all of the students begin unmatched, and the prices of the dorm rooms start at 0. In turns, unmatched students "bid" on their most preferred dorm room given the prices. The most recent bidder is the current (tentative) winner of the auction, and is (tentatively) matched to the room they bid on. If they are outbid, they go back to being unmatched. The algorithm halts when either there are no more unmatched students, or when none of the unmatched students are willing to bid on any of the rooms (because they are all too expensive). The algorithm has a parameter ϵ which is the "bid increment" — i.e. how much the price of a room rises each time there is a new winning bidder. This algorithm should remind you of the deferred acceptance algorithm for stable matchings you learned about in CIS 121.

Algorithm 1 The Ascending Price Auction with increment ϵ.

For all $v \in R$, set $p_v = 0$. For all $u \in L$, set $\mu(u) = \emptyset$.
while There exist any unmatched students **do**
  for Each unmatched student u **do**
    u "bids" on $v^* = \arg\max_v (w(u, v) - p_v)$ if $w(u, v^*) - p_{v^*} \geq \epsilon$. Otherwise, bidder u drops out. (and is "matched" to nothing):
    $u' = \mu^{-1}(v^*)$ is now unmatched: $\mu(u') = \emptyset$. Set $\mu(u) \leftarrow v^*$
    $p_{v^*} \leftarrow p_{v^*} + \epsilon$
  end for
end while
Output μ.

The first thing that we should prove about Algorithm 1 is that it is indeed an algorithm, meaning that it always halts and outputs *something*. Fixing an instance of a bipartite matching problem (L, R, w), we will write $W = \max_{u \in L, v \in R} w(u, v)$ to denote the largest weight in the instance. Note that for unweighted instances, by definition $W \leq 1$.

Lemma 5 *On any instance (L, R, w), Algorithm 1 with parameter ϵ halts and outputs a matching after at most T steps for:*

$$T \leq \frac{nW}{\epsilon}$$

Proof We analyze the vector of "prices". Firt we claim that we always have that $\sum_{v \in R} p_v \leq n \cdot W$. To see this, note that once a vertex $v \in R$ becomes matched over the course of the algorithm, it never becomes unmatched (unlike vertices $u \in L$). Thus *unmatched* vertices $v \in V$ have price $p_v = 0$. It is also the case that for every v, $p_v \leq W$. This is because the "bidders" $u \in L$ only bid on items with price $p_v < w(u, v) - \epsilon \leq W - \epsilon$, and thus after the final bid on any item, its price is at most W. There can be at most n matched vertices $v \in R$, because each is matched to a unique $u \in L$, and $|L| = n$. This completes the first claim. Now, it suffices to observe that $\sum_{v \in R} p_v$ increments by exactly ϵ with every iteration of the algorithm, by construction. Since the vector of prices is initialized at 0, the result is that there can be at most $T \leq \frac{nW}{\epsilon}$ iterations before we would have $\sum_{v \in R} p_v > n \cdot W$, a contradiction to our initial claim. ■

So we know that Algorithm 1 at least outputs *some* matching μ. Now we need to argue that μ has high weight. To do this, we will again use the fictional prices p computed by the algorithm.

Definition 6 *A matching μ paired with a set of prices p jointly satisfy the ϵ-approximate* market clearing conditions *if:*

1. *For any unmatched vertex $v \in R$, $p_v = 0$. (Unmatched goods have price 0)*
2. *For any vertex $u \in L$, $w(u, \mu(u)) - p_{\mu(u)} \geq \arg\max_{v \in R \cup \{\emptyset\}} w(u, v) - p(v) - \epsilon$ (Each student is obtaining their ϵ-most preferred room given the prices).*

Lemma 7 *Consider the matching μ output by Algorithm 1, together with the vector of prices p at the time that the algorithm halts. μ and p jointly satisfy the ϵ-aprpoximate market clearing conditions.*

Proof To verify the first condition, observe that once a vertex $v \in R$ becomes matched, it never becomes unmatched. Therefore if v is unmatched in the final output, it must never have received a "bid" and hence its price was never incremented from its initialization of $p_v = 0$.

To verify the second condition, note that if $\mu(u) = v$, then u was the last bidder to bid on v by construction. At the time that u bid on v, we had by construction that $v = \arg\max_v(w(u, v) - p_v)$. Immediately afterwards, p_v was incremented by ϵ, and so we had that $v \geq \max_v(w(u, v) - p_v) - \epsilon$. Since p_v was not further modified (since there were no future bidders) and other prices are non-decreasing, this condition must also hold for the final output. ■

Lemma 8 *Any matching μ that can be paired with prices p that satisfy the ϵ-approximate market clearing condition must satisfy:*

$$\sum_{i=1}^{n} w(u, \mu(u)) \geq OPT - \epsilon n$$

Proof Let μ^* be the optimal matching satisfying $w(\mu^*) = \text{OPT}$. We will compare μ to μ^*. We know from the 2nd market clearing condition that for every $u \in L$ we have:

$$w(u, \mu(u)) - p_{\mu(u)} \geq w(u, \mu^*(u)) - p_{\mu^*(u)} - \epsilon.$$

Summing this condition over each $u \in L$ (with the convention that $p_\emptyset = 0$) we get:

$$\sum_{u \in L} w(u, \mu(u)) - \sum_{u \in L} p_{\mu(u)} \geq \sum_{u \in L} w(u, \mu^*(u)) - \sum_{u \in L} p_{\mu^*(u)} - \epsilon n = \text{OPT} - \sum_{u \in L} p_{\mu^*(u)} - \epsilon n$$

Grouping the terms representing prices, we have:

$$\sum_{u \in L} w(u, \mu(u)) \geq \text{OPT} - \epsilon n + \left(\sum_{u \in L} p_{\mu(u)} - \sum_{u \in L} p_{\mu^*(u)} \right)$$

But observe that by the first market clearing condition, $\sum_{u \in L} p_{\mu(u)} = \sum_{v \in R} p_v$ (because terms not appearing in the first sum take value 0). Hence

$$\left(\sum_{u \in L} p_{\mu(u)} - \sum_{u \in L} p_{\mu^*(u)} \right) \geq 0$$

which completes the proof. ■

All together, we have proven the following theorem:

Theorem 9 *On any instance (L, R, w) of the bipartite matching problem, after $T \leq \frac{nW}{\epsilon}$ iterations, Algorithm 1 outputs a matching μ such that:*

$$w(\mu) \geq OPT - \epsilon n$$

Note that this algorithm has a parameter ϵ that lets us trade off its running time with the accuracy of its solution, but it never seems to promise us an exactly optimal solution. However, lets consider the implications of this theorem in the special case of unweighted bipartite matching problems. We recall that in this case, $W \leq 1$. However note also that in this case, for any matching μ, $w(\mu)$ is integer valued and OPT $\leq n$. Hence, if we take $\epsilon < 1/n$, then the above theorem gives us that $w(\mu) \geq$ OPT $- c$ for some $c < 1$, which by the integrality condition on $w(\mu)$ means that $w(\mu) =$ OPT! We thus have the following theorem for unweighted matching problems:

Theorem 10 *On any instance (L, R, w) of the unweighted bipartite matching problem, if we set $\epsilon = \frac{n}{n^2+1}$ after $T \leq n^2 + 1$ iterations, Algorithm 1 outputs a matching μ such that:*

$$w(\mu) = OPT$$

.