CSC2420: Lecture 9

- Announcement: Will have additional questions for problems set today/tonight and will set due date.
- Todays agenda :
  - Online and random order model algorithms for bipartite matching.
  - More on the Random Order Model
  - Related (online) problems
  - Sublinear time algorithms

Online Bipartite matching (a special case of adwords)

- We return to online algorithms and algorithms in the random order model (ROM).
- One nice sequence of results begins with a randomized online algorithm for bipartite matching due to Karp, Vazirani and Vazirani (1990).
- The model is that we have a bipartite graph (*U,V*) and nodes in *U* enter online revealing all their edges. A deterministic greedy matching produces a maximal matching and hence a ½ approximation.
- It is easy to see that any deterministic online algorithm cannot be better than a ½ approx. even when the degree of every *u* in *U* is at most (equal) 2

The randomized ranking algorithm

- The algorithm chooses a random permutation of the nodes in *V* and then when a node *u* in *U* appears, it matches *u* to the highest ranked unmatched *v* in *V* such that (*u,v*) is an edge (if such a *v* exists).
- Aside: making a random choice for each *u* is still ½ approx.
- Equivalently, this algorithm can be viewed as a deterministic algorithm ("greedy" that always matches when possible breaking ties consistently) in the ROM model. That is, let *v_1, …,v_n* be any fixed ordering of the vertices and let the nodes in *U* enter randomly, then match each *u* to the first unmatched *v* in *V* according to the fixed order. To argue this, consider fixed orderings of *U* and *V*; the claim is that the matching will be the same whether *U* or *V* is entering online.

The KVV result and recent progress

- Theorem: Ranking provides a (1-1/e) approximation. Error in original analysis; alternative proof (and extension) by Goel and Mehta (2008), simplified (slightly weaker result) in Birnbaum and Mathieu. Recall that this positive result can be stated either as the bound for a particular deterministic algorithm in the stochastic ROM model, or the randomized Ranking algorithm in the (adversarial) online model.
- The (1-1/e) bound is essentially tight for any randomized online (i.e. adversarial input) algorithm. In the ROM model, Goel and Mehta state inapproximation bounds of ¾ (for deterministic) and 5/6 (for randomized) algorithms.
- Recent 2011 results (Karande, Mehta, Tripathi) in ROM model show that in the ROM model, Ranking achieves approximation at least .653 (beating 1-1/e) and no better than .727.

The Yao Principle.

- There is a basic technique, the Yao principle, that can be used to establish inapproximations (or time bounds) for randomized algorithms within a given class of algorithms. Namely, we are trying to find a bad (say adversarial) input of some size *n* for a randomized algorithm showing that the expected approximation (or time bound) is bad were the expectation is with respect to the randomization in the algorithm. To do so, we construct a distribution over inputs of size *n* and show that every deterministic algorithm (within the same class of algorithms) has a bad ratio of expected performance (or expected time) with the expectation now over over the input distribution.
- Moreover, if one chooses the worst case distribution of inputs, then this is the bound for the randomized algorithm. But we tend to just use one direction in the Yao Principle.

A simple example

- Suppose we want to establish a inapproximation for every randomized online (i.e. adversarial input) algorithm for bipartite matching. It turns out that the worst case example (for all input sizes) will always be a graph having a perfect matching.
- Consider the case of $n = 2$ nodes on each side of the graph. Up to isomorphism, there are three graphs having a perfect matching. For two of these graphs any greedy algorithm will also construct a perfect matching. So we consider the remaining graph (the one causing problems for any deterministic online algorithm) . We construct a distribution for the input sequence. The first node in u has edges (u,v) , (u,v'). The second node u' has an edge (u',v) or (u',v') each with probability ½. Easy to show **E**[ALG] = ¾ and this bound is tight as the Ranking algorithm is a ¾ approximation on 2 by 2 bipartite graphs.

Sketch of the Ranking approximation (1-1/e) bound

- I am following the presentation in Birnbaum and Mathieu which I think is a slightly weaker result in that it shows a ratio for *n* by *n* bipartite graphs of $(1/n)\sum_{s=1}^{n}(1-1/(n+1))^s$ which limits to the stated (1-1/e) from below. I think the correct bound would replace *n+1* by *n* in the summation and then approach 1-1/e from above. This would then exactly match the ¾ bound for *n* = 2. But not clear that such a proof would work.

Analysis continued

- Let x_t denote the probability (over the random permutations of the vertices in V) that the vertex of rank *t* is matched. Then the main (not tight) lemma is: $1 - x_t \leq (1/n) \sum_{s=1}^{t} x_s$ Given this lemma, the result proceeds as follows:
- We can assume we are only considering graphs with perfect matchings and we can then obtain a bound by minimizing the x_t and hence setting the inequality to an equality. The recurrence can be expressed as $S_t(1 + 1/n) = 1 + S_{t-1}$ where S_t is $\sum_{s=1}^{t} x_s$. Solve for and divide S_n by *n* for bound

Analysis continued

- The recurrence solution is $S_t = \sum_{s=1}^{t}(1 - 1/(n+1))^s$ for all *t*.
- Setting *t = n* and then dividing by *n* gives the stated bound on the approximation ratio.
- The proof of the main lemma is somewhat subtle. The rough idea is as follows: given a random permutation *p* (in an instance of Ranking), take a random *v* in *V* out of *p* and place it at rank *t* to form permutation *p'*. If *v* is matched by *u* in a given perfect matching *m** but *v* is not matched in *p'*, then *u* is matched to a *v'* of rank at most *t* in *p*'. If *R_t* is defined as the set of all vertices in *U* matched to a vertex of rank at most *t* wrt *p*, then conditioned on *p*, the event *u* in *R*_t has probability |R_t|/n.

Getting past the (1-/1e) bound

- The ROM model can be considered as an example of what is called stochastic optimization in the OR literature. There are other stochastic optimization models that are perhaps more natural, namely i.i.d sampling from known and unknown distributions.
- Feldman et al (FOCS 2009) study the known distribution case and show a randomized algorithm that first computes an optimal offline solution (in terms of expectation) and uses that to guide an online allocation. They achieve a .67 approximation (improved to .699 Bahmani and Kapralov ESA 2010) and also show that no online algorithm can achieve better than 26/27 ~ .99 (improved to .902 in BK10).

Returning to the ROM model

- Karande, Mehta and Tripathi (STOC 2011) show that a c-approximation in the ROM model implies a c-approximation in the i.i.d. unknown distribution model. Furthermore they show that Ranking in the ROM model achieves approximation .653 and that Ranking is no better than .727.
- This can be contrasted with the previously mentioned Goel and Mehta lower bounds for any deterministic(3/4) and randomized algorithms (5/6) in the ROM model.

Extensions of (online) bipartite matching

- In the (single slot) adwords problem, the nodes in *U* are queries and the nodes in *V* are advertisers. For each query *q* and advertiser *i,* there is a bid *b*_{*q,*i} representing the value of this query to the advertiser. Each advertiser also has a hard budget *B_i* which cannot be exceeded. The goal is to match the nodes in *U* to *V* so as to maximize the sum of the accepted bids without exceeding any budgets. Without budgets, the problem then is edge weighted bipartite matching. In the online case, when a query arrives, all the relevant bids are revealed.

Results for adwords problem

- Here we are just considering the combinatorial problem and ignoring game theoretic aspects of the problem.
- The problem has been studied for the special (but well motivated case) that all bids are small relative to the budgets. (As such this problem is incomparable to the matching problem where all bids are in {0,1} and all budgets are 1.) Mehta et al (FOCS 2005) provide a deterministic online algorithm achieving the 1-1/e bound and show that this is optimal for all (randomized) online algorithms (i.e. adversarial input).

Greedy for the adwords problem (and bipartite and b-matching)

- Goel and Mehta (SODA2008) define a class of adwords problems which include the case of small budgets, bipartite matching and *b*-matching (where all budgets are equal to some *b* and all bids are equal to 1).
- For this class of problems, they show that a deterministic greedy algorithm achieves the familiar 1-1/e bound in the ROM model. Namely, the algorithm assigns each query (node in *U*) to the advertiser who values it most (truncating bids to keep them within budget and consistently breaking ties). Recall that Ranking can be viewed as greedy (with consistent tie breaking) in the ROM model.

Vertex weighted bipartite matching

- Aggarwal et al (SODA 2011) consider a vertex weighted version of the online biparitite matching problem. Namely, the vertices in *v* in *V* all have a known weight *w_v* and the goal is now to maximize the weighted sum of matched vertices in *V* where again vertices in *U* arrive online. This problem can be shown to subsume the adwords problem when all bids *b_{q,i}* = *b_i* from an advertiser are the same.
- It is easy to see that Ranking can be arbitrarily bad when there are arbitrary differences in the weight. Greedy (taking the maximum weight match) can be good in such cases. Can two such algorithms be somehow combined?
- Surprisingly, Aggarwal et al are able to achieve the same 1-1/e bound.

The vertex weighted online algorithm

- Perturbed-Greedy
  - For each *v* in *V*, pick *x_v* randomly in [0,1]
  - Let f(*x*) = 1 – e^{*1-x*}
  - When *u* in *U* arrives, match *u* to the unmatched *v* (if any) having the highest value of *w_v* * f(*x_v*). Break ties consistently.
- In the unweighted case when all w_v are identical this is the Ranking algorithm.

Lots of open problems ROM model

- The adwords problem without any restriction
- Beating 1-1/e for the vertex weighted or b-matching problems.
- In general any online problem can be studied with respect to this model.
- The first prominent use of this model is for the secretary problem; namely selecting the maximum element in a randomly ordered sequence. Here again 1-1/e is the best approximation. This has been generalized to the matroid secretary problem by Babaioff, Immorlica and Kleinberg in the context of mechanism design.

New topic: sublinear time algorithms

- An algorithm is *sublinear time* if its running time is o(n). As such an algorithm must provide an answer without reading the entire input. Thus to achieve non-trivial tasks, we almost always have to use randomness in sublinear time algorithms to sample inputs
- Example of a determinstic sublinear time algorithm. Suppose we are given a finite metric space M (with say n known points x_i) where the input is given as n^2 distance values d(x_i,x_j). The problem is to compute the diameter D of the metric space, that is, the maximum distance between any two points.
- For this maximum diameter problem, there is a simple O(n) time (and hence sublinear) algorithm; namely, choose an arbitrary point x' and compute D' = max_x d(x,x'). By the triangle inequality, D' is a 2-approximation of the diameter.

Finding an element in an (anchored) sorted linked list.

- Suppose we have an array A[i] for 1 <= i <= n where each A[i] is a pair (x_i, p_i) with x_1 = min{x_i} and p_i being a pointer to the next smallest value in the linked list. That is, x_{p_i} = min {x_j | x_j > x_i}. (For simplicity we are assuming all x_j are distinct.) We would like to determine if a given x occurs in the linked list an if so, output I such that x = x_i.
- The algorithm proceeds as follows:

  Let R = {j_i} be sqrt n randomly chosen indices plus the index 1. Access these A[j_i] to determine k such that x_k is the largest of the accessed array elements less than or equal to x.

  Then search forward 2 sqrt n steps on the linked list
- Claim: This is a one-sided error algorithm that will fail to return i such that x = A[i] with probability at most 1/7.
- Proof idea: Think of the points being sorted in the real line and look at the 2 sqrt n points starting at x and then decreasing. We succeed whenever these 2 sqrt n points contain one of the x_{j_i}.

Estimating average degree in a graph (example of use of Chernoff bounds)

- Given a graph G = (*V, E*) and |*V* | = *n*, we want to estimate the average degree *d* of all vertices of G. We want to construct an algorithm that approximates the average degree within a factor less than (2 + eps) with probability at least 3/4 in time O(n/poly(eps)). We will assume that we can access the degree *d_i* of any vertex *v_i* in one step.
- Like a number of results in this area, the algorithm is simple but the analysis requires a little care.
- The algorithm samples 8/eps random subsets *S_i* of *V* each of size (say) sqrt *n*/eps^3 computing the average degree *a_i* of nodes in each *S_i*. The output is the minimum of these {*a_i*}.

Analysis of approximation

- Since we are sampling subsets to estimate the average degree, we should expect that the estimates can be too high or too low but we will show that with high probability these estimates will not be too bad. More precisely,, we need:
- Lemma 1: $Prob[a_i < \frac{1}{2}(1-\epsilon)d] \leq \frac{\epsilon}{64}$
- Lemma 2: $Prob[a_i > (1+\epsilon)d] \leq 1 - \frac{\epsilon}{2}$