CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
Lecture 10

Allan Borodin

March 17, 2016

Announcements and todays agenda

- Announcements
  1. I have two sets of slides (and expecting a third set) for last weeks guest lecture (Lecture 8) by Aleksander Nikolov on the use of linear discrepency in algorithm design. I hope to have them posted by this weekend.
  2. Undergraduate Theory Group Talk by Aleksandar Nikolov - see next slide
  3. Assignment 2 due next Thursday, March 24. There are two small clarifications.
  4. I will begin posting questions soon (by the weekend) for Assignment 3. It looks like graduate grades are not due until May but I will have to check with the undergraduate office regarding when those grades are due. That will determine the due date(s).
- Todays agenda
  1. Continue with some discussion of extensions of (and questions about) online bipartite matching.
  2. Random walks and k-Sat

Undergraduate Theory Group seminar

Speaker: Aleksandar Nikolov, University of Toronto
Title: A Brief to Introduction to Discrepancy Theory
Abstract: Discrepancy theory is an area of mathematics that studies how well discrete objects can approximate continuous ones. In this talk we will introduce some of the main questions of the theory. We will see how low discrepancy point sets can be used to evaluate complicated integrals, and how to construct such point sets using balanced colorings. We will then mention computational questions about discrepancy, and, if time permits, briefly mention how the same balanced coloring problem can be used in designing approximation algorithms for NP-hard problems.

Extensions of online bipartite matching

- Weighted online matching
- Adwords
- Some additional ROM model results for problems relating to online matching
- Stochastic matching
- Online with Reassignments

The adwords problem: an extension of 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 i also usually 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 and when each advertiser will pay for at most one query, the problem then is edge weighted bipartite matching.
- In the online case, when a query arrives, all the relevant bids are revealed.

Some results for the adwords problem

- Here we are just considering the combinatorial problem and ignoring game theoretic aspects of the problem. That is, we are ignoring advertisers strategies for bidding. That is a major topic of interest in algorithmic game theory given the importance of online search engine bidding for slots.
- 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.
- For this small bid case, Mehta et al [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). Formally they assume that bids arrive online for each query q. When query q arrives, the bids $b_{q,i}$ for each advertiser i are revealed. The small bids assumption studies the approximation ratio as $\frac{\max b_{q,i}}{\min B_i} \to 0$ for all bidders (i.e. advertisers) i.

The relatively small bids case

The deterministic greedy algorithm for small bids

Let $f(x) = 1 - e^{-(1-x)}$
When q arrives, match q to the advertiser i maximizing $b_{q,i} \cdot f(T(i))$.
% Here $T(i) = \frac{b_{q,i}}{,} B_i$ is the fraction of the budget needed for this query
Break ties consistently, say by vertex ID.

- When all bids are in $\{0, 1\}$ and $B_i = B$ for all i, this is the B-matching problea;m that is, online bipartitie "matching" when every offline vertex can be matched to at most B online vertices.
- Kalyanasundaram and Pruhs show how to achieve a $1 - \frac{1}{e}$ approximation for B-matching as $B \to \infty$.
- The MSVV extension requires a more careful analysis of the tradeoff between bids and unspent budget. For their analysis they introduce a new technique using a "tradeoff revealing family of LPs" related to the "factor revealing LP" technique of Jain et al [2003] .

Greedy for a class of adwords problems

- Goel and Mehta [2008] define a class of adwords problems which include the case of small budgets, bipartite matching and B-matching.
- 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 (.e. 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.
- As such they provide what is perhaps the first published correct proof of the KVV result.

Vertex weighted bipartite matching

- Aggarwal et al [2011] consider a vertex weighted version of the ROM bipartite matching problem. Namely, the vertices $v \in V$ all have a known weight w_v and the goal is now to maximize the weighted sum of matched vertices in V when again vertices in U arrive online.
- This problem becomes the adwords problem when all bids $b_{q,i} = b_i$ from an advertiser i are independent of the query q. .
- 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? Aggarwal et al are able to achieve the same 1-1/e bound for this class of vertex weighted bipartite matching.
- Note the similarity to the small bids algorithm.

The vertex weighted online algorithm

The perturbed greedy algorithm

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, say by vertex ID.

In the unweighted case when all w_v are identical this is the Ranking algorithm.

The edge weighted algorithm in the ROM model

Kesselheim et al [ESA 2013] show how to extend the ideas of the ROM secretary algorithm to obtain a $\frac{1}{e}$ approximation to the edge weighted biparitite matching problem in the ROM model as well as extending this idea to set packing (i.e. combinatorial auctions).

Algorithm 1. Bipartite online matching

Input : vertex set R and cardinality $n = |L|$
Output: matching M

Let L' be the first $\lfloor n/e \rfloor$ vertices of L;
$M := \emptyset$;
for each subsequent vertex $\ell \in L - L'$ **do** `// steps` $\lceil n/e \rceil$ `to` n
  $L' := L' \cup \ell$;
  $M^{(\ell)}$:= optimal matching on $G[L' \cup R]$; `// e.g. by Hungarian method`
  Let $e^{(\ell)} := (\ell, r)$ be the edge assigned to ℓ in $M^{(\ell)}$;
  if $M \cup e^{(\ell)}$ is a matching **then**
    add $e^{(\ell)}$ to M;

[Kesselheim et al edge weighted biparitite matching algorithm]

Some additional remarks on "online bipartite matching"

- The ROM model subsumes the stochastic model where inputs are chosen i.i.d. from an unknown distribution (which in turn subsumes i.i.d. inputs from a known distribution). Why? Hence a positive result in the ROM model implies a positive result in the i.i.d. unknown distribution model.
- A research problem of current interest (work by Nicolas Pena) is to see to what extent some form of an extended online framework can yield a deterministic online bipartite matching algorithm with approximation ratio better than $1/2$.
- One can formulate the Buchbinder and Feldman method in the framework of the priority BT model of Alekhnovich et al. What is the best approximation acheievable by a *deterministic* (online or priority) "poly width" online algorithm BT algorithm?
- Pena shows that we cannot obtain a $\frac{1}{2} + \epsilon$ algorithm for any "Max-of k" online algorithm for $k = O(\log n / \log \log n)$.

Online algorithms allowing reassignments

There is a substantial history of results in scheduling allowing various forms of preemption. In the same spirit, we can allow online algorithms to undo previous decisions at some cost or in some limited way.
In particular,

1. As previously discussed, there is a constant approximation "greedy algorithm" for the weighted interval scheduling problem and the weighted JISP problem if previously accepted intervals can be deleted.
2. This revocable acceptance model can be applied to any packing problem where we are always maintaining a feasible solution.
3. Gupta et al [2014] consider the makespan problem in the restricted machines problem and show that when each job has size 1 (resp. arbitrary size), an assignment can be maintained that is within twice (resp. a factor $O(\log \log mn)$) of the optimal makespan while using amortized $O(1)$ *reassignments* per job.
4. This doesn't say anything about the maximum matching problem as to an algorithm that could tradeoff some reassignments for an improved approximation.

And (for now) some last thoughts relating to online algorithms

- In the online research area, there are various studies of "online algorithms with advice". There are two such models, one being that the online algorithm is initially given some small $\alpha(n)$ bits of advice bits, based on the entire input.
- While certain advice strings seem reasonable (e.g. if they can be computed easily), the framework here allows any advice string.
- Mikkelsen [2015] shows that no deterministic online algorithm with sublinear $o(n)$ advice can be substantially better than any online randomized algorithm without advice.
- It follows that we cannot obtain anything substantially better than a $(1 - \frac{1}{e})$ apprroximation for online unweighted bipartite matching with $o(n)$ advice.
- By a result of Bockenhauer [2011] there is an online algorithm using $O(\log n)$ advice that achieves a $(1 - \epsilon)(1 - \frac{1}{e})$ approximation for unweighted bipartite matching, thus matching what is the best possible approximation with $o(n)$ advice.

Last online thoughts continued

- Any online r bit advice algorithm immediately implies a non-uniform max-of-2^r online algorithm.
- Pena's inapproroximation about the limitation of max-of-k online algorithms for bipartite matching can then be used to show that $\Omega(\log \log n)$ advice is needed to achieve an approximation better than $\frac{1}{2} + \epsilon$.
- Pena's advice result improves upon a more restricted setting showing that $\Omega(\log \log \log n)$ advice is needed.
- These resuls show why it will be hard to prove inapproximation results about poly width online algorithms for bipartite matching.

Random walks and the random algorithm for 2-Sat and k-Sat

- First, here is the idea of the deterministic polynomial time algorithm for 2-Sat: We can first eliminate all unit clauses. We then reduce the problem to the directed $s - t$ path problem. We view each clause $(x \vee y)$ in F as two directed edges $(\bar{x}, y)$ and $(\bar{y}, x)$ in a graph G_F whose nodes are all possible literals x and $\bar{x}$. Then the formula is satisfiable iff there does not exist a variable x such that there are paths from x to $\bar{x}$ and from $\bar{x}$ to x in G_F.
- There is also a randomized algorithm for 2-SAT (due to Papadimitriou [1991]) based on a random walk on the line graph with nodes $\{0, 1, , n\}$. We view being on node i as having a truth assignment τ that is Hamming distance i from some fixed satisfying assignment τ^* if such an assignment exists (i.e. F is satisfiable).
- Start with an arbitrary truth assignment τ and if $F(\tau)$ is true then we are done; else find an arbitrary unsatisfied clause C and randomly choose one of the two variables x_i occurring in C and now change τ to τ' by setting $\tau'(x_i) = 1 - \tau(x_i)$.

The expected time to reach a satisfying assignment

- When we randomly select one the the two literals in C and complement it, we are getting close to τ^* (i.e. moving one edge closer to node 0 on the line) with probability at least $\frac{1}{2}$. (If it turns out that both literal values disagree with τ^*, then we are getting closer to τ^* with probability $= 1$.)
- As we are proceeding in this random walk we might encounter another satisfying assignment which is all the better.
- It remains to bound the expected time to reach node 0 in a random walk on the line where on each random step, the distance to node 0 is reduced by 1 with probability at least $\frac{1}{2}$ and otherwise increased by 1 (but never exceeding distance n). This perhaps biased random walk is at least as good as the case where we randomly increase or decrease the distance by 1 with probability equal to $\frac{1}{2}$.

Claim:

The expected time to hit node 0 is at most $2n^2$.

- To prove the claim one needs some basic facts about Markov chains.

The basics of finite Markov chains

- A finite Markov chain M is a discrete-time random process defined over a set of states S and a matrix $P = \{P_{ij}\}$ of transition probabilities.
- Denote by X_t the state of the Markov chain at time t. It is a memoryless process in that the future behavior of a Markov chain depends only on its current state: $Prob[X_{t+1} = j | X_t = i] = P_{ij}$ and hence $Prob[X_{t+1} = j] = \sum_i Prob[X_{t+1} = j | X_t = i] Prob[X_t = i]$.
- Given an initial state i, denote by r_{ij}^t the probability that the first time the process reaches state j occurs at time t;
  $r_{ij}^t = Pr[X_t = j \text{ and } X_s \neq j \text{ for } 1 \leq s \leq t-1 | X_0 = i]$
- Let f_{ij} the probability that state j is reachable from initial state i;
  $f_{ij} = \sum_{t>0} r_{ij}^t$.
- Denote by h_{ij} the expected number of steps to reach state j starting from state i (hitting time); that is, $h_{ij} = \sum_{t>0} t \cdot r_{ij}^t$
- Finally, the *commute time* c_{ij} is the expected number of steps to reach state j starting from state i, and then return to i from j; $c_{ij} = h_{ij} + h_{ji}$

Stationary distributions

- Define $\mathbf{q}^t = (q_1^t, q_2^t, \ldots, q_n^t)$, the state probability vector (the distribution of the chain at time t), as the row vector whose i-th component is the probability that the Markov chain is in state i at time t.
- A distribution π is a stationary distribution for a Markov chain with transition matrix P if $\pi = \pi P$.
- Define the underlying directed graph of a Markov chain as follows: each vertex in the graph corresponds to a state of the Markov chain and there is a directed edge from vertex i to vertex j iff $P_{ij} > 0$. A Markov chain is *irreducible* if its underlying graph consists of a single strongly connected component. We end these preliminary concepts by the following theorem.

Theorem: Existence of a stationary distribution

For any finite, irreducible and aperiodic Markov chain,

(i) There exists a *unique* stationary distribution π.

(ii) For all states i, $h_{ii} < \infty$, and $h_{ii} = 1/\pi_i$.

Back to random walks on graphs

- Let $G = (V, E)$ be a connected, non-bipartite, undirected graph with $|V| = n$ and $|E| = m$. A uniform random walk induces a Markov chain M_G as follows: the states of M_G are the vertices of G; and for any $u, v \in V$, $P_{uv} = 1/deg(u)$ if $(u, v) \in E$, and $P_{uv} = 0$ otherwise.
- Denote by $(d_1, d_2, \ldots, d_n)$ the vertex degrees. M_G has a stationary distribution $(d_1/2m, \ldots, d_n/2m)$.
- Let $C_u(G)$ be the expected time to visit every vertex, starting from u and define $C(G) = \max_u C_u(G)$ to be the *cover time* of G.

Theorem: Aleliunas et al [1979]

Let G be a connected undirected graph. Then

1. For each edge (u, v), $C_{u,v} \leq 2m$,
2. $C(G) \leq 2m(n-1)$.

- It follows that the 2-SAT random walk has expected time at most $2n^2$. to find a satisfying assignment in a satisfiable formula. Can use Markov inequality to obtain probability of not finding satisfying assignment.

Extending the random walk idea to k-SAT

- The random walk 2-Sat algorithm might be viewed as a drunken walk (and not an algorithmic paradigm). Or we could view the approach as a local search algorithm that doesn't know when it is making progress on any iteration but does have confidence that such an exploration of the local neighborhood is likely to be successful over time.
- We want to extend the 2-Sat algorithm to k-SAT. However, we know that k-SAT is NP-complete for $k \geq 3$ so our goal now is to improve upon the naive running time of 2^n, for formulas with n variables.
- In 1999, Following some earlier results, Schöning gave a very simple (a good thing) random walk algorithm for k-Sat that provides a substantial improvement in the running time (over say the naive 2^n exhaustive search) and this is still almost the fastest (worst case) algorithm known.
- This algorithm was derandomized by Moser and Scheder [2011].
- Beyond the theoretical significance of the result, this is the basis for various Walk-Sat algorithms that are used in practice.

Schöning's k-SAT algorithm

The algorithm is similar to the 2-Sat algorithm with the difference being that one does not allow the random walk to go on too long before trying another random starting assignment. The result is a one-sided error alg running in time $\tilde{O}[(2(1-/1k)]^n$; i.e. $\tilde{O}(\frac{4}{3})^n$ for 3-SAT, etc.

Randomized k-SAT algorithm

Choose a random assignment τ
Repeat 3n times % n = number of variables
If τ satisfies F then stop and accept
Else Else Let C be an arbitrary unsatisfied clause
  Randomly pick and flip one of the literals in C
End If

Claim

If F is satisfiable then the above succeeds with probability p at least $[(1/2)(k/k-1)]^n$. It follows that if we repeat the above process for t trials, then the probability that we fail to find a satisfying assignment is at most $(1-p)^t < e^{-pt}$. Setting $t = c/p$, we obtain error probability $(\frac{1}{e})^c$.