CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
An introductory (i.e. foundational) level graduate course.

Allan Borodin

November 8, 2012; Lecture 9

Brief Announcements

1. Game theory reading group meets at 4 in theory lab
2. Assignment 2 due next Thursday. I may add one more easy question.
3. We need to finish up the $1 - 1/e$ analysis for Yannakakis' IP/LP randomized rounding algorithm. This is needed for question 6 of the assignment.

The SDP/vector program approach: Max-2-Sat

- We briefly consider an important extension of the IP/LP approach, namely representing a problem as a strict quadratic program and then relaxing such a program to a vector program. Vector programs are known to be equivalent to semidefinite programs.
- For our purposes of just introducing the idea of this approach we will not discuss SDP concepts but rather just note that such programs (and hence vector programs) can be solved to arbitrary precision within polynomial time. This framework provides one of the most powerful optimization methods.
- We illustrate the approach in terms of the Max-2-Sat problem. A very similar algorithm and analysis produces the same approximation ratio for the Max-Cut problem.

The quadratic program for Max-2-Sat

- We introduce {-1,1} variables y_i corresponding to the propositional variables. We also introduce a homogenizing variable y_0 which will correspond to a constant truth value. That is, when $y_i = y_0$, the intended meaning is that x_i is set *true* and *false* otherwise.
- We want to express the $\{-1, 1\}$ truth value $val(C)$ of each clause C in terms of these $\{-1, 1\}$ variables.
  1. $val(x_i) = (1 + y_i y_0)/2$
     $val(\bar{x}_i) = (1 - y_i y_0)/2$
  2. If $C = (x_i \vee x_j)$, then $val(C) = 1 - val(\bar{x}_i \wedge \bar{x}_j) = 1 - (\frac{1-y_i y_0}{2})(\frac{1-y_j y_0}{2}) = (3 + y_i y_0 + y_j y_0 - y_i y_j)/4 = \frac{1+y_0 y_i}{4} + \frac{1+y_0 y_j}{4} + \frac{1-y_i y_j}{4}$
  3. If $C = (\bar{x}_i \vee x_j)$ then $val(C) = (3 - y_i y_0 + y_j y_0 + y_i y_j)/4$
  4. If $C = (\bar{x}_i \vee \bar{x}_j)$ then $val(C) = (3 - y_i y_0 - y_j y_0 - y_i y_j)/4$

The quadratic program for Max-2-Sat continued

- The Max-2-Sat problem is then to maximize $\sum w_k val(C_k)$ subject to $(y_i)^2 = 1$ for all i
- By collecting terms of the form $(1 + y_i y_j)$ and $(1 - y_i y_j)$ the max-2-sat objective can be represented as the strict quadratic objective: $\max \sum_{0 \leq i < j \leq n} a_{ij}(1 + y_i y_j) + \sum b_{ij}(1 - y_i y_j)$ for some appropriate a_{ij}, b_{ij}.
- Like an IP this integer quadratic program cannot be solved efficiently.

The vector program relaxation for Max-2-Sat

- We now relax the quadratic program to a vector program where each y_i is now a unit length vector $\mathbf{v}_i$ in $\Re^{n+1}$ and scalar multiplication is replaced by vector dot product. This vector program can be (approximately) efficiently solved (i.e. in polynomial time).
- The randomized rounding (from $\mathbf{v}_i^*$ to y_i) proceeds by choosing a random hyperplane in $\Re^{n+1}$ and then setting $y_i = 1$ iff $\mathbf{v}_i^*$ is on the same side of the hyperplane as $\mathbf{v}_0^*$. That is, if $\mathbf{r}$ is a uniformly random vector in $\Re^{n+1}$, then set $y_i = 1$ iff $\mathbf{r} \cdot \mathbf{v}_i^* \geq 0$.
- The rounded solution then has expected value $2\sum a_{ij} Prob[y_i = y_j] + \sum b_{ij} Prob[y_i \neq y_j]$; $Prob[y_i \neq y_j] = \frac{theta_{ij}}{\pi}$ where θ_{ij} is the angle between $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$.

The approximation ratio (in expectation) of the rounded solution

Let $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta)} \approx .87856$ and let OPT_{VP} be the value obtained by an optimal vector program solution.
Then **E**[rounded solution] $\geq \alpha \cdot (OPT_{VP})$.

The random walk algorithm for 2-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). 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 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 $(\frac{1}{e})^c$.

Randomized online bipartite matching and the adwords problem. NOTE: Not yet discussed in lecture.

- We return to online algorithms and algorithms in the random order model (ROM). Here we have already seen evidence of the power of randomization in the context of the MaxSat problem.
- Another nice sequence of results begins with a randomized online algorithm for bipartite matching due to Karp, Vazirani and Vazirani [1990]. We quickly overview some results in this area as it represents a topic of current interest. (The recent FOCS 2012 conference had a session of three papers related to this topic.)
- In the online bipartite matching problem, we have a bipartite graph G with nodes $U \cup V$. Nodes in U enter online revealing all their edges. A deterministic greedy matching produces a maximal matching and hence a $\frac{1}{2}$ approximation.
- It is easy to see that any deterministic online algorithm cannot be better than a $\frac{1}{2}$ approximation 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 only a $\frac{1}{2}$ approx.
- Equivalently, this algorithm can be viewed as a deterministic greedy (i.e. always matching when possible and breaking ties consistently) algorithm in the ROM model.
- That is, let $\{v_1, \ldots, v_n\}$ 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

KVV Theorem

Ranking provides a $(1 - 1/e)$ approximation.

- Original analysis is not rigorous.
- There is an alternative proof (and extension) by Goel and Mehta [2008], and then another proof in Birnbaum and Mathieu [2008].
- Recall that this positive result can be stated either as the bound for a particular deterministic algorithm in the stochastic ROM model, or as the randomized Ranking algorithm in the (adversarial) online model.
- KVV show that 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 $\frac{3}{4}$ (for deterministic) and $\frac{5}{6}$ (for randomized) algorithms.
- In the ROM model, Karande, Mehta, Tripathi [2011] show that Ranking achieves approximation at least .653 (beating $1 - 1/e$) and no better than .727.

Some comments on the Birnbaum and Mathieu proof

- The worst case example a (n, n) graph with a perfect matching.
- In particular, for $n = 2$, the precise expected competitive (i.e. approximation) ratiois $\frac{3}{4}$. The inapproximation can be seen by using the Yao principle for obtaining bounds on randomized algorithms.

The main lemma in the analysis

Let x_t be the probability (over the random permutations of the vertices in V) that the vertex of rank t is matched. Then $1 - x_t \leq \frac{1}{n} \sum_{s=1}^{t} x_s$

- Letting $S_t = \sum_{s=1}^{t} x_s$ the lemma can be restated as $S_t(1 + 1/n) \geq 1 + S_{t-1}$ fo all t. Given that the graph has a perfect matching, the expected competitive ratio is S_n/n. It is shown that $\frac{1}{n} S_n \geq 1 - (1 - \frac{1}{n+1})^n \to 1 - 1/e$.

Getting past the $(1 - 1/e)$ 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 [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 [2010] and also show that no online algorithm can achieve better than 26/27 .99 (improved to .902).
- Karande, et al [2011] how that an approximation in the ROM model implies thre same approximation in the unknown distribution model. They show that the KVV Ranking algorithm achieves approximation .653 in the ROM model and is no better than .727.

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 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.
- 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).

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 (i.e. when 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 (.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.

Vertex weighted bipartite matching

- Aggarwal et al [2011] consider a vertex weighted version of the online 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 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 for this class of vertex weighted bipartite matching.

The vertex weighted online algorithm

The perturbed greedy algorithm

For each $v \in V$, pick r_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.

Some open problems in the ROM model

There are many open problems in the ROM model. In general any online problem can be studied with respect to this model. For example, relevant to what we have just discussed:

- The adwords problem without any restriction
- Beating $1 - 1/e$ for the vertex weighted or b-matching problems.
- Perhaps, the first prominent use of this model is for the secretary problem; namely selecting the maximum element (or best k elements) 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. For arbitrary matroids, the approximation ratio remains an open problem.