CSC2420: Lecture 8

- Todays agenda :
- Comment on randomness and complexity theory
- Summary of last lecture discussion of recent work on combinatorial algorithms for MaxSat.
- Submodular Max sat
- Random walks and k-SAT

The why of randomization

- As mentioned last class, there are some problem settings (simulation, cryptography, sublinear time algorithms) where randomization is necessary. Beyond that:
- We have been using randomization to improve the approximation ratio (so far just for Max-Sat).
- We have also seen the conceptual power of randomization even when a given alg can be derandomized.
- In complexity theory the big question is how much can randomization lower the complexity of a problem. For decision problems, there are three polynomial time randomized classes ZPP (zero-sided), RP (1-sided) and BPP (2-sided) error. The big question (and conjecture?) is BPP = P?

Some problems not known to be poly time but known to be in randomized poly time (see Wigderson slides) .

- The symbolic determinant problem.
- Solving a quadratic equation in $Z_p[x]$ for a large prime p.
- Given n, find a prime in $[2^n, 2^{n+1}]$
- Estimating volume of a convex body given by a set of linear inequalitiies.

What are the limits to such combinatorial algorithms for MaxSat?

- Apparent contradictions: The Slack algorithm is a randomized online algorithm achieving approx ¾.
- Azar, Gamzu, Roth (ESA 11) show that no randomized algorithm can obtain an approx ratio better than 2/3.
- As observed in Azar et al, the negative result is for the weakest (priority) input model and the Slack algorithm is implemented in the middle model.
- Poloczek (ESA 11) shows that no deterministic priority algrorithm (in the middle model) can achieve the randomized ¾ ratio. In a strong sense this show that (unlike the naïve randomized algorithm), the Slack algorithm cannot be derandomized.

Locals search for MaxSat

- I am not sure what is known about local search algorithms for (arbitrary) MaxSat; i.e. can a local search algorithm can achieve the bounds achieved by Johnson's algorithm and recent extensions.
- For exact Max-*k*-Sat, Khanna, Motwani, Sudan and U. Vazirani (1994) consider the oblivious Hamming distance local search algorithm for weighted exact Max-*k*-Sat. Namely, given a current truth assignment tau, search a Hamming neighbourhood (of distance say *r*) for an improved solution. They show that for *r = 1,* the algorithm achieves a *k/k+1* fraction of the total weight and that this is the locality gap even when *r = o*(*n*); in particular this is 2/3 for exact Max-2-Sat whereas we can achieve ¾ for the more general MaxSat by the randomized Slack algorithm.
- However, they also show that a non-oblivious Hamming distance 1 algorithm achieves the Johnson (2^k-1)/2^k bound for exact Max-*k*-Sat. What is the best deterministic or randomized local search algorithm (combinatorial algorithm) for MaxSat?

Submodular MaxSat

- The main result in Azar et al is that the monotone submodular MaxSat problem can be approximated with ratio 2/3 (in the weakest model).
- Submodular MaxSat is a equivalent to maximizing a submodular function subject to a binary partition matroid. In the one direction we need, think of the universe U as being {*a_1,b_1*, …,*a_n,b_n*} where each pair {*a_i,b_i*} is a block of the partition and *a_i* (resp. *b_i*) corresponds to setting *x_i* true (resp. false). We then think of the submodular function *g* (on sets of clauses) as a function *f* on the subset *S* of *U* chosen (subject to choosing at most one element from each pair). Namely, any such subset S determines a set *t*(*S*) of satisfied clauses and then *f = g* (*t*(*S*)). It is easy to see that g monotone submodular implies f is also.

The Submodular MaxSat algorithm

- The algorithm is a randomized version of the natural (local) greedy algorithm for maximizing a submodular function subject to a partition matroid.
- Namely, let $f\_S(a) = f(S+a) - f(a)$ denote the marginal gain when adding a to S. Since we will only be adding one of a_t and b_t, we can choose which to use proportional to their gain.
- Since this will turn out to be an online algorithm, we won't care about how ordering of the pairs $\{(a\_t, b\_t)\}$

The algorithm

- Algorithm 1. Proportional Select
- Input: A monotone submodular function $f : 2^X \rightarrow R_+$ and a binary partition matroid M
- Output: A set $S \subseteq X$ approximating the maximum of f under constraint M
- 1: $S_0 \leftarrow \emptyset$
- 2: for $t \leftarrow 1$ to n do
- 3: $w_t \leftarrow f_{S_{t-1}}(a_t) + f_{S_{t-1}}(b_t)$
- 4: $p_{a_t} \leftarrow f_{S_{t-1}}(a_t)/w_t$
- 5: $p_{b_t} \leftarrow f_{S_{t-1}}(b_t)/w_t$
- 6: Pick s_t at random from $\{a_t, b_t\}$ with respective probabilities (p_{a_t}, p_{b_t})
- 7: $S_t \leftarrow S_{t-1} \cup \{s_t\}$
- 8: end for

Approx bound for Proportional Select

- Let S be the algorithm solution. Then E[f(s)] >= (2/3) OPT. (For the value oracle input model, there is an offline information-theoretic inapproximation of ¾ due to Mirrokni, Schapira and Vondrak.)
- Sketch of proof: We consider the best extension possible OPT_i of the solution S_i at end of the ith iteration. The goal is to show that (in expectation) the loss L_i in the value of OPT_i is bounded by half of the gain in f(S_{i-1}); that is, we have the lemma: E[L_i} = OPT_{i-1} – OPT_i <=

  (1/2) E[f(S_{i-1}+ s_i)-f(S_{i-1})]

Proof of appox ratio continued

- Sum_*i* *L_i* = OPT_0 – OPT_*n* = OPT – *f*(*S*) which combined with the lemma yields the desired result.
- The lemma is proven using the following:
  - If the algorithm chooses the *s_i* (say *b_i*) not in OPT_{*i-1*}, then using submodularity: OPT_{*i-1*}-OPT_*i* <= f(*S_i*)-f(*S_*{*i-1*}+ *a_i*)
  - Combing this with the probabilistic choice of *s_i* shows: **E**[*L_i*|*S_*{*i-1*}] <= m(*i-1,a_i*)m(*i,b_i*)/m(*i-1,a_i*) +m(*i-1,b_i*) where m(*i-1,s_i*) is the marginal gain adding s_i to S_{i-1}
  - The prob. choice of *s_i* also shows **E**[m(*i-1,s_i*)|*S_*{*i-1*}] = (m(*i-1,a_i*))^2 +(m(*i-1,b_i*))^2/ m(*i-1,a_i*) +m(*i-1,b_i*)
  - The expected los to gain ratio is then the ratio which is <=m(*i-1,a_i*)m(*i,b_i*)/ (m*(i-1,a_i*))^2 +(m(*i-1,b_i*))^2<=1/2

Random walk algorithm for 2-Sat

- As mentioned before, there is a deterministic polynomial time algorithm for 2-Sat. (The idea is to view each clause (*x* v *y*) in *F* as two directed edges (*x',y*) and (*y',x*) in a graph *G_F* whose nodes are all possible literals *x* and *x'*. Then the formula is satisfiable iff there does not exist a variable x such that there are paths *x* to *x'* and *x'* to *x* in *G_F*.
- There is also a randomized algorithm (Papadimitriou 1991) based on a random walk on the line graph with nodes *0,1,…,n*. We view being on node *i* as being Hamming distance *i* from some fixed satisfying assignment tau if such an assignment exists (i.e. *F* is satisfiable).
- The expected time to hit node 0 is at most *n*^2.

Hitting times, commute times and cover times for uniform random walks

- Here is a quick summary of what we need to know about Markov chains and random walks.
- Finite Markov Chain $M = (X,P)$ Concepts
  - P_{ij} = Pr[i -> j] = $Pr[X_{t+1} = j \mid X_t = i]$ indep of history
  - h_{ij} = expected hitting time to reach state j from i.
  - c_{ij} = expected commute time = $h_{ij} + h_{ji}$
  - $q^t = (q^t_1, \ldots, q^t_n)$ is state distribution at time t
  - q is a stationary distribution if $qP = q$
  - Theorem: Any finite, irreducible, aperiodic Markov chain M has 1) a unique stationary distribution q and
    2) for all states X_i, $h_{ii} = 1/q_i$ (i.e. finite)

Uniform random walk as Markov chain (Aleliunas, Karp, Lipton,Lovasz, Rackoff)

- Let $G = (V,E)$ be connected undirected graph; define $P_{uv} = 1/d_u$ if (u,v) in E; o.w. 0 ($d_u = \deg(u)$)
- If G is not bipartite then it satisfies conditions for unique stationary distribution. (Bipartite case can be dealt with by considering 2 step process.)
- Let $C_u(G)$ denote cover time = expected time to have visited all nodes starting at u and let C(G) be $\max_u C_u(G)$ be the *worst case cover time.*
- $c_{u,v} <= 2m$ *and* $C(G) <= 2m(n-1)$; $n = |V|$, $m = |E|$

Schoening's *k*-Sat algorithm

- In 1999, Schoening gave a very simple (and that is a good thing) randomized local search algorithm for *k*-Sat that provides a substantial improvement in the running time (over say the naïve *2^n* exhaustive search) and this is still almost the fastest algorithm known. This algorithm was recently "fully derandomized" in 2011 by Moser and Scheder. 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 O*[(2(1-/1k))^n*] e.g. O*[(4/3)^*n*] for 3-SAT. O* ignores poly(n) factors

The *k*-Sat algorithm (*k* > 2)

- Choose a random assignment tau
  Repeat 3*n* times % *n* = number of variables
    If tau satisfies F then stop and accept
  Else Let C be an arbitrary unsatisfied clause
  Randomly pick and flip one of the literals in C
- 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*, this a very small failure prob.

Estimating success probability p

- Let tau* be some satisfying assignment and like the 2-Sat algorithm, the analysis is about finding tau* and it can be that the algorithm succeeds well before that in finding a different satisfying assignment. The initial random assignment picks an assignment that is Hamming distance away from tau* with probability (*n* choose *j*) 2^{-*n*}.
- Now conditioned on the random assignment being distance j, a simple argument shows that the probability q_j of reaching tau* is at least (*1/k*)^*j*. A little more care is needed to show that *p* is at least [1/(*k-1*)]^*j*.

Finishing the analysis

- The more careful analysis considers the possibility that in reaching tau* the random walk takes i steps in the wrong direction so then needs *j+2i* steps in the right direction and we consider this possibility for all *i <= j* (and hence the reason for bounding the walk to 3*n* steps). It can then be show that *q_j* is at least [1/(*k-1*)]^*j*.
- Putting this together with the probability that the starting assignment is *j* away from tau* yields the desired probability.