CSC 373 Lecture 23

Review from lecture 22:

- Return to local search
- The basic local search format
- Mention of oblivious vs non-oblivious local search

Today

- (Weighted) Max Cut
- Exact Max-k-SAT (from Khanna et al, 1999)

The basic local search meta-algorithm

- Initialize *S*
  While there is a "better solution" *S'* in *Nbhd(S)*
  S := S'
  End While
- Here "better" can mean different things. For a search problem, it can mean "closer" to being feasible (in some sense); for an optimization problem it usually means being an improved solution.

The weighted max cut problem

- Let $G = (V,E)$ be a graph with non-negative edge weights. In the (weighted) max cut problem, the goal is to find a cut so as to maximize the cardinality (resp. weight) of the cut.
- As in min cuts, a cut is a partition (A,B) of the vertices and the weight of the cut (what we called the capacity in the max flow-min cut setting) is the sum of weights of edges (u,v) such that u in A, v in B.
- There is a simple local search algorithm that achieves approximation ratio 2 and it is still an open problem if any greedy-like or local search algorithm can do better than this ratio.

Single move local search

- Let *(A,B)* be a partition. (Note that in this problem every partition is feasible.) Then *N_d(A,B)* (i.e. say as denoted by the characteristic vector of *A*) is the neighbourhood of partitions (i.e. char. vectors) at distance at most d from *(A,B)*.
- Choose any initial partition *(A,B)*

While there is a better partition *(A',B')* in *N_1(A,B)*
(A,B) : = (A',B')

End While

The locality (totality) gap

- The single move algorithm provides a 2 approximation; that is, when the algorithm terminates, the value of any (global optimal) solution will be at most twice the of the weight of the local search solution (i.e a local optimum).
- In fact, if W is the sum of all edge weights, then any local optimum (A,B) has value at least W/2.
- This kind of ratio is called the absolute ratio or the totality ratio.)

Proof of totality gap

- WLG, say *G* is a clique by setting any missing edge weight to 0.
- Given a local opt, we have for all *u in A*:

sum_{v in A} w(u,v) <= sum_{v in B} w(u,v)

or else *u* can be moved to *B*.

Summing over *all u in A*

2 sum_{u,v in A} w(u,v)

<= sum_{u \in A, v \in B} w(u,v)

= w(A,B)

Proof continued

- We repeat the argument for set B to obtain
- *2 sum_{u,v in B} w(u,v)*
  <= sum_{u in A, v in B} w(u,v) = w(A,B)
- Adding these inequalities and dividing by 2, we get *sum_{u,v in A} w(u,v) + sum_{u,v in B} w(u,v)*

  <=w(A,B)

  Adding *w(A,B)* to both sides we get

  *W <= 2 *w(A,B)*

Running time of the algorithm

- The algorithm clearly terminates since there are only finitely (but unfortunately exponentially) many partitions.
- KT say it is an open problem if there is a way to find a local optimum in polynomial time. But one can achieve a ratio as close as we want to 2 in polynomial time by looking for a solution *(A'B')* which is sufficiently better; namely *w(A'B')* is >= *(1+epsilon/n) w(A,B)* for arbitrarily small epsilon.
- Modified alg achieves *2 * (1 + epsilon)* approx. in *(n/epsilon) log W* iterations.

Final comment on this algorithm

Using the neighburhood *N_d(S)* for any constant *d* (or even sublinear *o(n)* Hamming distance) will not essentially improve the bound.

It is an open problem if there is any “simple combinatorial algorithm” that can do substantially better than this very simple local search algorithm.

Exact Max-2-Sat

- Let *F* be an exact 2 CNF formula;
  F = C_1 ^ C_2 … ^C_m

  where *C_i = (ell_i^1 v ell_i^2) and ell_i^j* is a literal in *{x_k, bar x_k | 1 <= k <=n}* .
- In the weighted version, each *C_i* has a weight *w_i.*
- The goal is to find tau so as to maximize *w(F|tau)*, the weighted sum of satisfied clauses.
- This is a constraint satisfaction problem

The natural oblivious local search

- A natural oblivious local search algorithm uses a Hamming distance d neighbourhood
- *N_d(tau) = {tau': tau* and *tau'* differ on at most d variables }

Choose any initial truth assignment *tau*

While there exists a truth assignment tau in *N_d(tau*)

such that *W(tau') > W(tau)* set *tau := tau'*

End While

NB: in what follows I will switch to approx ratio < 1.

How good is this algorithm?

- It can be shown that for d = 1, the totality ratio for this local search algorithm is 2/3 (and more generally for exact Max-k-Sat the ratio is *k/(k+1)*. This ratio is a tight ratio for any d = o(n).
- This is in contrast to a naive greedy algorithm derived from a randomized algorithm that achieves totality ratio *(2^k-1)/2^k*.
- ("In practice" the local search algorithm actually performs better than the naive greedy and one could always start with the greedy algorithm and then apply local search.

Analysis of oblivious local search for exact max-2-sat

- Let *tau* be a local optimum and let *S_0* (resp. *S_1, S_2*) be those clauses that are not satisifed (resp. satisfied by exactly one literal, by two literals) by *tau* and let *W(S_i)* be the corresponding weight .
- Let *A_j* (resp. *B_j*, resp. *C_j*) be those clauses containing the variable *x_j* such that no literal (resp. exactly the one literal involving x_j, both literals) in any clause in *S_0* (resp *S_1, S_2*) is satisfied by *tau*.
- Since *tau* is a local optimum, the weight of these clauses must satisfy *W(A_j) <= W(B_j)*. Summing over all variables *x_j, 2 W(S_0) <= W(S_1*) noting that each clause in *S_0* gets counted twice.

Finishing the analysis

- It follows then that the ratio of clause weights not satisfied to the sum of all clause weights is
  W(S_0)/ (W(S_0_+W(S_1) + W(S_2)
  <= W(S_0) /(3S_0) + W(S_2)
  <= W(S_0)/3W(S_0)
- It is not easy to verify but there are examples showing that this 2/3 bound is essentially tight for any *N_d* neighbourhood for *d = o(n)* and it is also claimed that the bound is at best 4/5 whenever *d < n/2*. For *d = n/*2 the algorithm would be optimal.

Using the proof

- As in some previous examples (e.g. distinct edge weights imply a unique MST), we can learn something from this proof to improve the performance.
- Note that we are not using anything about *W(S_2)*. If we could guarantee that *W(S_0)* was at most *W(S_2)* then the ratio of clause weights not satisfied to all clause weights would be 1/4 .
- We can do this by enlarging the neighbourhood to include *tau'* = the complement of *tau*.

The non oblivious local search

- Now we return to the idea previously mentioned that clauses in *S_2* are more valuable than clauses in *S_1* (because they are able to withstand any single variable change).
- The idea then is to weight *S_2* clauses more heavily. Specifically, in each iteration we attempt to find a *tau'* in *N_1(tau)* that improves the potential function *3/2W(S_1) + 2W(S_2)* instead of the oblivious *W(S_1) + W(S_2)*.

Sketch of ¾ totality bound for this non oblivious local search

- Let *P_i,j* (resp. *N_i,j*) be the weight of all clauses in *S_i* containing *x_j* (resp bar *x_j*).
- Here is the key observation for a local optimum tau wrt the stated potential:
  -(1/2) P_2,j – (3/2) P_1,j
  <= (1/2) N_1,j + (3/2) N_0,j
- Summing over variables *P_1 = N_1 = W(S_1)*, *P_2 = 2 W(S_2)* and *N_0 = 2 W(S_0)* and using the above inequality we obtain
  3W(S_0) <= W(S_1) + W(S_2)