CSC373S Lecture 20

- Continuing from the last lecture, we are considering a very natural local search algorithm for the MIS problem on $k+1$ claw-free graphs (in the hope of improving the k approximation bound tht one can obtain from a greedy algorithm). As stated in the problem set a basic oblivious local search for the weighted MIS problem (restricted to $k+1$ claw-free graphs is the following:

  $S := \emptyset$
  While $\exists u \in V - S$ such that $w_u > w(N(u) \cap S)$
  $\quad S := (S - N(u)) \cup \{u\}$
  End While

  For an unweighted graph this simplifies to a greedy algorithm :

  $S := \emptyset$
  While $\exists u \in V - S$ such that $S \cup \{u\}$ is an independent set
  $\quad S := S \cup \{u\}$
  End While

- For the unweighted case the proof that the locality gap (i.e. the ratio between any local optimum and a global optimum) is k (and hence that the above local search produces a k-approximation) is a charging argument, the same argument that one could use to show that the greedy algorithm is a k-approximation. Your assignment question requires a somewhat more involved charging argument for the weighted case. For a given graph G let S be any locally optimal solution (which in the unweighted case means any maximal independent set) and let A be an arbitrary solution (and in particular an optimal solution).

  Lets define a function $h : A \to S$ which is a k to 1 mapping which then implies $|A| \leq k \cdot |S|$. First for any vertex $v \in S \cap A, h(v) = v$. (So without loss of generality lets assume that $S \cap A = \emptyset$.) Now consider any $u \in A$ and (having arbitrarily ordered all vertices) define $h(u) =$ the smallest vertex $v \in S$ such that $(u, v) \in E$. Clearly there must be such a vertex v or else the local search algorithm would have added u to S. Now the claim is that at most k different vertices $u \in A$ can be mapped to the same v. This follows immediately from the $k+1$ claw-free property since A is supposed to be an independent set and $h(u) = v$ implies that $(u, v) \in E$ so that more than k such vertices u would imply a bigger claw (with center v).

- As such the above local search algorithm isn't very interesting as it is really just another way of restating the greedy algorithm. But we can slightly generalize this algorithm to obtain an improved approximation for the unweighted case. Namely, we will now consider larger sets of vertices in $V - S$ as possible improvements. Letting $N_S(I) = \{u \in S : \exists x \in I \text{ with } (u, x) \in E\}$, we have the following algorithm for any t:

$S := \emptyset$
While there exists an independent set $I \subseteq V - S$ such that $|I| \leq t$ and $|I| > |N_S(I)|$ then
$S := (S \cup I) - N_S(I)$
End While

Such a set I is called a t-improvement and we can call this a t-local-search algorithm. We state (without proof for the time being) the following result:

For $t = 2$, the locality gap is $\frac{k+1}{2}$. The algorithm can be implemented so as to run in "nearly linear" time. The approximation bound is $\frac{5}{3}$ for $k = 3$. For $k \geq 4$, the approximation bound can be improved (by choosin larger t) to $\frac{k}{2} + \epsilon$ but taking time $O(n^{\log_k 1/\epsilon})$. So the message here is that oblivious local search can essentially cut the approximation ratio in half but that is all it can do.

- Unfortunately (or interestingly, if you study algorithms), this kind of oblivious t local-search does not provide the same improvement in the weighted case. Here is what is known for the *weighted* MIS problem restricted to $k + 1$ claw-free graphs:

  1. The oblivious t local-search algorithm has a locality gap of $k - 1 + \frac{1}{t}$ so basically larger neighbourhoods do not improve the locality gap.
  2. If instead of taking $S = \emptyset$ as the initial set, if we initially take S to be the solution of the greedy algorithm and then use the following oblivious local-search algorithm, the approximation bound is $\frac{2}{3}k$. Here (as above) $N_S(C) = \{u \in S : \exists x \in C \text{ with } (u, x) \in E\}$

     While there is a vertex $v \in S$ and a claw C centered at v such that $w(C) > w(N_S(C))$ then
     $S := (S \cup C) - N_S(C)$
     End While

     This shows that while the locality gap does not improve, the approximation ratio does improve by taking the greedy solution as the initial solution.
  3. There is a non-oblivious local search method that achieves approximation ratio $\frac{k+1}{2} + \epsilon$. After creating a maximal independent set, the method looks for a claw that improves the related "potential function" $w^2(S) = \sum_{v \in S} w^2(v)$. What is the intuition for using such a potential function? Suppose we had two roughly equal valued solutions, one having many nodes with small weights and one having few nodes with large weights. Which is the better solution for hopefully finding a better solution in a Hamming local neighbourhood? Intuitively, the solution S havng few nodes seems like the right choice as few nodes in $V - S$ seem likely to be adjacent. This intution leads one to try using a potential function such as $w^2(S)$ which favours small sets of nodes with large weights.

- Another example of a very basic and natural local search algorithm is for the (weighted) max cut problem. See KT, section 12.4. Let $G = (V, E)$ be a graph

with edge weights $w : E \rightarrow \Re^{\geq 0}$. 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 $\sum_{(u,v)\in E} w(u, v)$. While finding a (global) min cut or an $s-t$ cut is poly time computable, max cut is an NP hard problem and the best approx ratio known for a polynomial time approximation algorithm is .878 (or $1/.878 \approx 1.139$) so as to consistently stay with approximation ratios bigger than 1) using semi-definite programming SDP. There is some evidence (based on a different complexity assumption) that this SDP based approximation is the best approximation possible for a polynomial time algorithm. Recently there there has been interest in obtaining a "combinatorial algorithm" (one not relying on solving LPs or SDPs) that is better that a 2-approximation. That has been accomplished but still this remains a topic of interest. There is a simple to state 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.

In this algorithm, we consider the single move neighbourhood $N(A, B)$ of a partition (i.e. feasible solution) (A, B); that is, $N(A, B)$ is the set of n partitions (A', B') that can be obtained by moving one node from A to B (or from B to A). If we let χ_A be the n bit characteristic vector of A, then $N_d(A, B)$ is the neighbourhood of vectors at distance d from χ_A. (When feasible solutions are subsets, a small Hamming distance ball is often used as the neighbourhood.)

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

Claim: This is a 2 approximation algorithm; that is, when the algorithm terminates, the value of any (global optimal) solution will be at most twice the of the value of a local optimum.

In fact, if $W = \sum_{e\in E} w_e$, then any local optimum (A, B) has value $\geq W/2$. (This is called the absolute ratio and also called the totality ratio.)

Proof: To simplify notation, WLG, say G is a clique with all edges present by setting any missing edge weight to 0. Given a local opt, we have :

for all $u \in A$, $\sum_{v\in A} w_(u, v) \leq \sum_{v\in B} w(u, v)$
or else u can be moved to B.

summing over all $u \in A$

$2\sum_{u,v\in A} w(u, v) \leq \sum_{u\in A,v\in B} w(u, v) = w(A, B)$

We repeat the same argument for the set B to obtain
$2\sum_{u,v\in B} w(u,v) \leq \sum_{u\in A, v\in B} w(u,v) = w(A,B)$

Adding these inequalities and dividing by 2, we get $\sum_{u,v\in A} w(u,v) + \sum_{u,v\in B} w(u,v) \leq w(A,B)$

And finally, adding $w(A,B)$ to both sides we get $W \leq 2w(A,B)$

Note: The above local search uses the distance 1 neighbourhood $N_1(S)$. It turns out that any contant distance d (using the neighburhood $N_d(S)$) will not improve the bound.

- So any local optimum here is within a factor of 2 of being optimal. But how long does it take for this algorithm to terminate? The algorithm clearly terminates since there are only finitely (but unfortunately exponentially) many partitions. The following discussion illustrates a common method to overcome possible exponential time for termination.

  KT say it is an open problem if there is a way to find a local optimum in polynomially many steps. But as they point out, one can achieve a ratio as close as we want to 2 in polynomially many steps. More specifically, instead of looking for a better solution in $N(A,Ba)$, we look for a solution $(A'B')$ which is sufficiently better, namely $w(A'B') \geq (1+\epsilon/n)w(A,B)$ for ϵ as small as we want.

  Claim: For any $\epsilon > 0$, any approximate local optimum $(A'B')$ for this modified local search algorithm satisfies $w(A'B') \geq 1/(2+\epsilon) \cdot W$

  Proof: We add $(2\epsilon/nw(A,B)$ to the right side of each inequality; eg for all $u \in A, \sum_{v\in A} w(u,v) \leq \sum_{v\in B} w(u,v) + 2\frac{\epsilon}{n} w(A,B)$

  Claim: For any $\epsilon > 0$, if we start with say an (A,B) solution so that the largest weight edge is in the cut, then the modified algorithm terminates in $(n/\epsilon)logW$ local neighbourhood searches.

  Note: In the weighted MIS local search results, we started the local search from a solution already known to have some reasonable approximation and that could be used to speed up the convergence.

- The final local search application I will discuss now is for the exact Max-2-Sat which we already discussed in the context of a simple randomized algorithm and how that algorithm can be de-randomized by the method of conditional expectations thus becoming a greedy online algorithm. While the bounds given here for local search are not as good as what can be obtained by other methods, this application serves to again demonstrate the usefulness of non-oblivious local search.

Let F be an exact 2 CNF formula $F = C_1 \wedge C_2 \ldots \wedge C_m$ where $C_i = (\ell_i^1 \vee \ell_i^2)$ and $\ell_i^j \in \{x_k, \bar{x}_k | 1 \leq k \leq n\}$ is a literal. In the weighted version, each C_i has a weight w_i.

For a truth assignment, let $W(\tau)$ be the weighted sum of satisfied clauses in $F(\tau)$.

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

Choose any initial truth assignment τ
While there exists a truth assignment $\tau' \in N_d(\tau)$ such that $W(\tau') > W(\tau)$
    $\tau := \tau'$
End While

Note: In the following, I am switching to totality and approximation bounds that are less than 1 as this seems to be traditional for this problem.

It can be shown that for $d = 1$, the absolute (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 $d = o(n)$. That is, even very large Hamming neighbourhoods do not improve the 2/3 absolute ratio. This is in contrast to the greedy algorithm (that is derived from the randomized algorithm) that achieves abosolute ratio $(2^k - 1)/2^k$. So the oblivious local search algorithm does not have a good worst case absolute ratio. (In practice the local search algorithm actually performs better than this naive greedy and one could always start with the greedy and then apply local search.)

Here is what I again find interesting. As I said, even if we take the neighbourhood to be $N_d(n)$ with $d(n) = o(n)$, the ratio essitally does not improve. (This is just a comment about this local search algorithm.) In contrast here is a better non-oblivious local search algorithm.

Let $W_0(\tau), (respW_1(\tau)$ and $W_2(\tau))$ be the weighted sum of clauses satisfied by 0 (resp 1 , 2) literals.

Informal claim: all things being equal, better to satisfy a clause with two literals than one literal. So we introduce the following potential function: $(3/2)W_1(\tau) + 2W_2(\tau)$

Claim: Using this potential function with Hamming distance 1, the absolute ratio is now 3/4 (and more generally $(2^k - 1)/2^k$ for exact Max-k-Sat).