CSC 373: Algorithm Design and Analysis
Lecture 22

Allan Borodin

March 13, 2013

Announcements and Outline

Announcements

- Lecture this Friday

Today's outline

- Almost new topic: start or rather return to local search
- Max Cut
- Exact Max-2-Sat

Local search: the other conceptually simplest approach for solving search/optimization problems

We now begin a discussion of the other (than greedy) conceptually simplest search/optimization algorithm, namely local search.

The vanilla local search paradigm

1: Initialize S
2: **while** there is a "better" solution S' in "$Nbhd(S)$" **do**
3: $\quad S := S'$
4: **end while**

If and when the algorithm terminates, the algorithm has computed a **local optimum**. To make this a precise algorithmic model, we have to say:

1. How are we allowed to choose an initial solution?
2. What consititutes a local neighbourhood $Nbhd(S)$?
3. What do we mean by "better"?

Answering these questions (especially as to defining local neighbourhood) will often be quite problem specific.

Towards a precise definition for local search

On choosing an initial solution

- We clearly want the initial solution to be efficiently computed
- And to that end (so as to be precise) we can (for example) say that the initial solution is a random solution, or a greedy solution or adversarially chosen.
- Of course, in practice we can use any efficiently computed solution and this is done in practice.
- For an optimization problem, we usually begin with a feasible initial solution. For a search problem, we will (necessarily) start with a non-feasible solution.
- We will focus on local search for optimization problems.

Choosing the local neighbourhood

We want the local neighbourhood $Nbhd(S)$ to be such that we can efficiently search for a "better" solution (if one exists).

1. In many problems, a solution S is a subset of the input items or equivalently a $\{0,1\}$ vector, and in this case we often define the $Nbhd(S) = \{S' \mid d_H(S, S') \leq k\}$ for some small k where $d_H(S, S')$ is the Hamming distance.
2. More generally whenever a solution is a vector over a small domain D, we can use Hamming distance to define a local neighbourhood. Hamming distance k implies that $Nbhd(S) = |D|^k$. Hence if necessary, the neighbourhood can be exhaustively search for a better solution.
3. We can view Ford Fulkerson flow algorithms as local search algorithms where the neighbourhood of a solution S (i.e. a flow) are flows obtained by adding an augmenting path flow. This is an exponential size neighbourhood but one that can be searched efficiently.

What does "better" solution mean? Oblivious and non-oblivious local search

- For a search problem, we would generally have a non-feasible initial solution and "better" can then mean "closer" to being feasible.
- For an optimization problem it usually means being an improved solution which respect to the given objective. For reasons I cannot understand, this has been termed *oblivious* local search.
- For some applications, it turns out that rather than searching to improve the given objective function, we search for a solution in the local neighbourhood that improves a related potential function and this has been termed *non-oblivious* local search.
- And in searching for an improved solution, we may want an arbitrary improved solution, a random improved solution, or the best improved solution in the local neighbourhood.
- For efficiency we may insist that there is a "sufficient" improvement.

The weighted max cut problem

- Our first local search algorithm will be for the (weighted) max cut problem that we formalized in our discussion of IPs.

The (weighted) max-cut problem

- Given a (undirrected) graph $G = (V, E)$ and in the weighted case the edges have non negative weights.
- **Goal:** Find a partition (A, B) so as to maximize the size (or weight) of the cut $E' = \{(u, v) | u \in A, v \in B, (u, v) \in E\}$.

- We can think of the partition as a characteristic vector χ in $\{0, 1\}^n$ where $n = |V|$. Namely, say $\chi_i = 1$ iff $v_i \in A$.
- Let $N_d(A, B) = \{(A', B') |$ the characteristic vector of the cut (A', B') is Hamming distance at most d from the characteristic vector for $(A, B)\}$
- So what is a natural local search algorithm for (weighted) max cut?

A natural oblivious local search for weighted max cut

Single move local search for weighted max cut

```
Initialize (A, B) arbitrarily
while there is a better partition (A′, B′) ∈ N_1(A, B) do
    (A, B) := (A′, B′)
end while
```

- This single move local search algorithm is a $\frac{1}{2}$ approximation; that is, when the algorithm terminates, the value of the computed local optimum will be at least half of the (global) optimum value.
- In fact, if W is the sum of all edge weights, then $w(A, B) \geq \frac{1}{2} W$.
- This kind of ratio is sometimes called the absolute ratio or totality ratio and the approximation ratio must be at least this good.
- The worst case (over all instances and all local optima) of a local optimum to a global optimum is called the locality gap.
- It may be possible to obtain a better approximation ratio than the locality gap but the approximation ratio is at least as good as the locaity gap.

Proof of totality gap for the max cut single move local search

- The proof is based on the following property of any local optimum:

$$\sum_{v \in A} w(u,v) \leq \sum_{v \in B} w(u,v) \text{ for every } u \in A$$

- Summing over all $u \in A$, we have:

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

- Repeating the argument for B we have:

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

- Adding these two inequalites and dividing by 2, we get:

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

- Adding $w(A,B)$ to both sides we get the desired $W \leq 2w(A,B)$.

The complexity of the single move local search

- **Claim:** The local search algorithm terminates on every input instance.
  - Why?

- Although it terminates, the algorithm could run for exponentially many steps.

- It seems to be an open problem if one can find a local optimum in polynomial time.

- However, we can achieve a ratio as close to the state $\frac{1}{2}$ totality ratio by only continuing when we find a solution (A', B') in the local neighborhood which is "sufficiently better". Namely, we want

$$w(A', B') \geq (1 + \epsilon) w(A, B) \text{ for any } \epsilon > 0$$

- This results in a totality ratio $\frac{1}{2(1+\epsilon)}$ with the number of iterations bounded by $\frac{n}{\epsilon} \log W$.

Final comment on this local search algorithm

- It is not hard to find an instance where the single move local search approximation ratio is $\frac{1}{2}$.
- Furthermore, for any constant d, using the local Hamming neighbourhood $N_d(A, B)$ still results in an approximation ratio that is essentially $\frac{1}{2}$. And this remains the case even for $d = o(n)$.
- It is an open problem as to what is the best "combinatorial algorithm" that one can achieve for max cut.
- As previously mentioned there is a vector program relaxation of a quadratic program that leads to a .878 approximation ratio.

Exact Max-2-Sat

- **Given:** An exact 2-CNF formula

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

where $C_i = (\ell_i^1 \vee \ell_i^2)$ and $\ell_i^j \in \{x_k, \bar{x}_k \,|\, 1 \leq k \leq n\}$.

- In the weighted version, each C_i has a weight w_i.

- **Goal:** Find a truth assignment τ so as to maximize

$$W(\tau) = w(F \,|\, \tau),$$

the weighted sum of satisfied clauses w.r.t the truth assignment τ.

The natural oblivious local search

- A natural oblivious local search algorithm uses a Hamming distance d neighbourhood

$$N_d(\tau) = \{\tau' \mid \tau \text{ and } \tau' \text{ differ on at most } d \text{ variables}\}$$

Oblivious local search for Exact Max-2-Sat

1: Choose any initial truth assignment τ
2: **while** there exists $\hat{\tau} \in N_d(\tau)$ such that $W(\hat{\tau}) > W(\tau)$ **do**
3: $\quad \tau := \hat{\tau}$
4: **end while**

How good is this algorithm?

- Note: in what follows I will use approximation ratios < 1.
- It can be shown that for $d = 1$, the approximation ratio is $\frac{2}{3}$.
- In fact, for every formula, the algorithm finds an assignment τ such that $W(\tau) \geq \frac{2}{3}\sum_{i=1}^{m} w_i$, the weight of all clauses, and we say that the "totality ratio" is at least $\frac{2}{3}$.
- (More generally for Exact Max-k-Sat the ratio is $\frac{k}{k+1}$).
- This ratio is essentially 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 often performs better than the naive greedy and one could always start with the greedy algorithm and then apply local search.

Analysis of the oblivious local search for Exact Max-2-Sat

- Let τ be a local optimum and let
  - S_0 be those clauses that are not satisfied by τ
  - S_1 be those clauses that are satisfied by exactly one literal by τ
  - S_2 be those clauses that are satisfied by two literals by τ

  Let $W(S_i)$ be the corresponding weight.

- We will say that a clause involves a variable x_j if either $x_j$or $\bar{x}_j$ occurs in the clause. Then for each j, let
  - A_j be those clauses in S_0 involving the variable x_j.
  - B_j be those clauses C in S_1 involving the variable x_j such that it is the literal x_j or $\bar{x}_j$ that is satisfied in C by τ.

  Let $W(A_j), W(B_j)$ be the corresponding weights.

Analysis of the oblivious local search (continued)

- Summing over all variables x_j, we get
  - $2W(S_0) = \sum_j W(A_j)$ noting that each clause in S_0 gets counted twice.
  - $W(S_1) = \sum_j W(B_j)$
- Given that τ is a local optimum, for every j, we have

$$W(A_j) \leq W(B_j)$$

  or else flipping the truth value of x_j would improve the weight of the clauses being satisfied.

- Hence (by summing over all j),

$$2W_0 \leq W_1.$$

Finishing the analysis

- It follows then that the ratio of clause weights not satisfied to the sum of all clause weights is

$$\frac{W(S_0)}{W(S_0)+W(S_1)+W(S_2)} \leq \frac{W(S_0)}{3W(S_0)+W(S_2)} \leq \frac{W(S_0)}{3W(S_0)}$$

- It is not easy to verify but there are examples showing that this $\frac{2}{3}$ bound is essentially tight for any N_d neighbourhood for $d = o(n)$.

- It is also claimed that the bound is at best $\frac{4}{5}$ whenever $d < n/2$. For $d = n/2$, the algorithm would be optimal.

- In the weighted case, as in the max-cut problem, we have to worry about the number of iterations. And here again we can speed up the termination by insisting that any improvement has to be sufficiently better.