CSC 373: Algorithm Design and Analysis
Lecture 26

Allan Borodin

March 22, 2013

Announcements and Outline

Announcements

- Lecture this Friday; slower pace for rest of term.
- Next week in tutorial, we will go over all the basic probability concepts that are needed for this part of the course.
- There are many texts which will have a short section on basic probability concepts; for example, see section 13.12 of the Kleinber and Tardos text.

Today's outline

- Finish up weighted set cover with randomized rounding
- The de-randomizing of the naive randomized Max-Sat into a greedy algorithm
- Return to some previous topics
- Greedy and local search algorithms for $k+1$ clawfree graphs.

Set cover IP/LP for Weighted Set Cover and randomized rounding

- There is a very natural and efficient greedy algorithm for solving the weighted set cover problem with approximation H_d where $d = max_i|S_i|$. What would you try?
- Recall that $O(H_m) = O(\log m)$, where m is the size of the universe.
- But we want to use this problem to give a final example of IP and randomized rounding.
- Note that in the randomized Max-Sat algorithms, we never had to worry about whether or not a solution was feasible since every truth assignment is feasible. The only issue was the approximation ratio.
- The following randomized algorithm will with high probability produce a cover that is within a factor $O(H_d) = O(\log m)$ of the optimum.
- This is also an opportunity to (re)introduce a little more probability.

The IP/LP randomized rounding

An IP formulation of weighted Set Cover

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} w_i x_i \\
\text{s.t.} \quad & \sum_{i\,:\,j \in S_i} x_i \geq 1 && \text{for each } j = 1, \ldots, m \\
& x_i \in \{0, 1\} && \text{for each } i = 1, \ldots, n
\end{aligned}
$$

- We relax this 0/1 IP by replacing the integrality constraints $x_i \in \{0, 1\}$ by the following constraints:

$$
0 \leq x_i \qquad \text{for each } i = 1, \ldots, n
$$

- We solve this LP and find an optimal solution $\{x_1^*, \ldots, x_n^*\}$.
- We know $x_i^* \leq 1$ since in an optimal solution, each x_i^* is at most 1.
- Thus, we can treat the x_i^* values as probabilities and choose S_i (to be in our set cover) with probability x_i^*.

Some comments on this randomized rounding

- This is a covering problem and the sets produced by randomized rounding will most likely not be a cover.

- So we will have to repeat this process enough times to have a good probability that all elements are covered.

- We will next show:

This randomized rounding algorithm with high probability produces a cover whose cost is within a factor $O(\log m)$ of the optimum.

The analysis

- It is easy to calculate the expected cost of the "partial cover" C' of sets selected by the LP optimum.

- Namely,

$$\mathbb{E}[cost(C')] = \sum w_i \cdot \mathbb{P}[S_i \text{ is chosen}] = \sum w_i x_i^* = \text{OPT-LP}$$

- Now we need to calculate the probability that a given $u \in U$ is not covered.

- Let's say that u occurs in sets $S_1, \ldots, S_k$. The LP solution must satisfy the constraint:

$$\sum_{i \,:\, u \in S_i} x_i^* \geq 1$$

The analysis continued

- Under this constraint, we can maximize the probability that u is not covered by $x_i^* = 1/k$ for $1 \leq i \leq k$. Thus,

$$\mathbb{P}[\text{ } u \text{ is not covered }] \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$$

- Suppose now that we run the same randomized rounding algorithm $c \ln m$ times, where $m = |U|$, for some constant c. On each iteration, add sets (given by the rounded LP) to the set cover.

- While we may be adding the same set many times (and hence overcounting),

$$\text{the cost of the "cover"} \leq (c \ln m) \cdot \text{OPT-LP}.$$

- Thus,

$$\mathbb{P}[\text{ } u \text{ is not covered }] \leq \left(\frac{1}{e}\right)^{c \ln m} = \left(\frac{1}{m}\right)^c.$$

Finishing the analysis

The union bound

Let $R_1, \ldots, R_m$ be a set of random events with $\mathbb{P}[R_j] \leq p_j$. Then

$$\mathbb{P}[\text{ at least one } R_j \text{ occurs }] \leq p_1 + \ldots + p_m$$

- Let R_j be the event that element j is not covered. Then by the union bound,

$$\mathbb{P}[\text{ some } u \in U \text{ is not covered }] \leq |U| \left(\frac{1}{m}\right)^c = \left(\frac{1}{m}\right)^{c-1}$$

- Using the Markov inequality we can show that the expected cost is within $O(\log m) \cdot \text{OPT-LP}$ with good probability. Hence, with good probability we get a cover with cost $O(\log m) \cdot \text{OPT-LP}$.

- This certainly shows that with good probability we get a cover with cost $O(\log m) \cdot \text{OPT}$ since $\text{OPT-LP} \leq \text{OPT}$.

De-randomizing the naive Max-Sat algorithm

- We recall the naive randomized algorithm that we used for the Exact Weighted Max-k-Sat problem.

- In that algoirhtm we randomly and indepedently set each propositional variable x_i so that $\mathbb{P}[x_i = \mathit{true}] = \mathbb{P}[x_i = \mathit{false}] = \frac{1}{2}$.

- The expected weight of the solution is $\frac{2^k - 1}{2^k} \sum_j w_j$.

- By using the method of conditional expectations, the algorithm can be de-randomized.

- We wish to make this more explicit.

Johnson's algorithm is the derandomized algorithm

- Yannakakis [1994] presented the naive algorithm and showed that Johnson's algorithm is the derandomized naive algorithm.
- Yannakakis also observed that for arbitrary Max-Sat, the approximation of Johnson's algorithm is at best $\frac{2}{3}$.
  - For example, consider the 2-CNF $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge \bar{y}$ when variable x is first set to true.
- Chen, Friesen, Zheng [1999] showed that Johnson's algorithm achieves approximation ratio $\frac{2}{3}$ for arbitrary weighted Max-Sat.
- For arbitrary Max-Sat (resp. Max-2-Sat), the current best approximation ratio is 0.797 (resp. 0.931) using semi-definite programming and randomized rounding.

Understanding Johnson's algorithm as an online greedy algorithm

- The randomized algorithm (and hence its de-randomized counterpart) is an online algorithm in the sense that we can set the variables in any order.
  - Can view this as an online algorithm where the algorithm will set the variables in the order given without full knowledge of the entire formula.
- To make things a little more precise, we will say that a propositional variable is represented by the names of the clauses in which it appears positively and the names of the clauses in which it appears negatively.
- In addition, we specify the number of literals in each of these clauses.

Johnson's algorithm continued

- The method of conditional expectations tell us that

$$\mathbb{E}[W_F] = \frac{1}{2} \cdot \mathbb{E}[W_F \mid x_1 = \mathit{true}] + \frac{1}{2} \cdot \mathbb{E}[W_F \mid x_1 = \mathit{false}]$$

- Therefore at least one of these two assignments to x_1 must gives the desired expectation.

- **Important observation:** We can decide which assignment of x_1 by computing the expectations knowing the sign of x_1 and number of literals in each clause to which it belongs. But is there a more efficient way to do this and one that does not involve looking at the entire formula?

Johnson's algorithm continued

- Let's see more explicitly how to set each variable x_i.
- Let's say (without loss of generality by renaming) that x_i occurs positively in some clause C_j in which there are k_j literals.
- We consider the expected weight to be lost from clause C_j if we set the variable false. Namely, we will remove x_i from C_j and therefore will lose $w_j \frac{1}{2^{k_j}}$ since we will be decreasing the expected weight of that clause from $w_j \frac{2^{k_j}-1}{2^{k_j}}$ to $w_j \frac{2^{k_j-1}-1}{2^{k_j-1}}$.
- Of course, if we set $x_i = \mathit{true}$, then we have satisfied C_j and no longer need to consider that clause.
- We then sum these loses for each clause in which x_i occurs for each of the two possible truth values and take the best choice.

Johnson's Max-Sat Algorithm [1974]

For all clauses C_i, let $w_i' := w_i/(2^{|C_i|})$
Let L be the set of clauses in F and X the set of variables
For $x \in X$ (or until L empty)
    Let $P = \{C_i \in L \mid x \text{ occurs positively in } C_i\}$
    Let $N = \{C_j \in L \mid x \text{ occurs negatively in } C_j\}$
    If $\sum_{C_i \in P} w_i' \geq \sum_{C_j \in N} w_j'$
        % that is, we have more to lose setting $x := \mathit{false}$
        $x := \mathit{true}$
        $L := L - P$
        For all $C_r \in N$, $w_r' := 2w_r'$ **End For**
    Else
        $x := \mathit{false}$; $L := L - N$
        For all $C_r \in P$, $w_r' := 2w_r'$ **End For**
    End If
    Delete x from X
End For

Returning to some previous topics

- We will spend the remaining lectures reviewing some previous topics and, if time permits, add some additional material (but which will not be part of the test/exam material).
- We will start by introducing the following optimization problem:

The weighted set *packing* problem

- As in the set cover problem, we are given a collection of sets $\mathcal{C} = \{S_1, S_2, \ldots, S_n\}$ over a universe $U = \{e_1, e_2, \ldots, e_m\}$ and $w_i = w(S_i)$.
- Goal: Choose a subcollection $\mathcal{C}'$ of disjoint sets so as to *maximize* $\sum_{i:S_i \in C'} w_i$.
- When the size of the sets S_i is restricted such that $|S_i| \leq k$ we call this the k-set packing problem.

Set packing (SP) as an MIS problem

- The graph theoretic interpretation of the above problem is as follows: Given a set packing instance define graph $G = (V, E)$ with $V = \{S_1,\ S_2,\ \ldots S_n\}$ and $E = \{(S_i, S_j) | S_i \bigcap S_j \neq \varnothing\}$.
- The (weighted) set packing problem becomes the (weighted) maximum independent set (W)MIS problem on this graph.
- **Note:** This is another example of a polynomial time transformation:
$$\text{SP} \leq_p \text{MIS}.$$
- Given an MIS problem, interpreting it as a set packing problem is also quite straightforward.
    - The set of elements $U = e_1,\ e_2,\ \ldots, e_m$ consist of the edges of the graph $G = (V, E)$.
    - The collection of sets S_i for $1 \leq i \leq |V| (= n)$ is given by the adjacency list of the vertices.
    - Note that in this case, $m \leq n^2$ and $|S_i| \leq n$.
- That is, this second transformation shows that MIS $\leq_p$ SP.

Brief discussion of randomized polynomial time complexity classes

- ZPP is a randomized "0-sided error" analogue of P, always giving the correct answer (for a decision problem $L \in \mathrm{ZPP}$) but running in *expected* polynomial time (rather than deterministic polynomial time).

- The randomized algorithm given for the symbolic determinant problem (and more generally for testing polynomial identities) is a "1-sided polynomial time randomized algorithm" always halting in polynomial time but possibly making an error (on one side) with sufficiently low probability. Such algorithms define the class RP.

- Not hard to see that $\mathrm{ZPP} = \mathrm{RP} \cap \mathrm{coRP}$ and $\mathrm{ZPP} \subseteq \mathrm{RP} \subseteq \mathrm{NP}$.

- There is also a 2-sided error analogoue complexity class called BPP.