CMPUT 675: Approximation Algorithms **Winter 2005**

Lecture 8: Feb 7

Lecturer: Mohammad R. Salavatipour *Scribe: Xiaozhen Niu*

SAT is perhaps the most well-known NP-complete problem. There are several variations of this problem that have been studied. Today and the next lecture, we will consider an optimization version of SAT which is called Max-SAT.

8.1 Max-SAT

The Max-SAT problem is defined as follows:

Input: A boolean formula τ over variables $x_1, \ldots, x_n$ in CNF which has clauses $C_1, \ldots, C_M$, each clause C_j has a weight $w_j \geq 0$.

Question: Find a truth assignment to the variables that maximizes the total weight of satisfied clauses.

Special cases: if all the weights w_j are 1 (unit-weight) then we essentially have to maximize the number of satisfied clauses. The Max-k-SAT problem is the restriction of Max-SAT to the instances in which every clause has at most k literals. The Max-Ek-SAT is the restriction of the problem to the instances in which every clause has *exactly* k literals.

Theorem 8.1 *Max-k-SAT is NP hard for any $k \geq 2$.*

Note that 2-SAT is polynomially solvable and k-SAT (for $k \geq 3$) is NP-hard.

Today, we will see 3 approximation algorithms for Max-SAT. The first one is good when the sizes of clauses are large. Then we show how to improve upon this algorithm. The third algorithm (seen next lecture) will be good when the clauses are small. At the end we show how the combination of first and third algorithm yields a better approximation algorithm.

8.1.1 Simple Randomized Algorithm

This is perhaps the most obvious randomized (and maybe the dumbest possible randomized) algorithm. Flip a fair coin for every variable (independently) to choose the value True or False for that variable, i.e. set it to True/False with probability of $\frac{1}{2}$ and return this truth assignment. We call this algorithm **Alg1** for Max-SAT.

Theorem 8.2 (Johnson'74) *Alg1 is a $\frac{1}{2}$-approximation for Max-SAT.*

Proof: Let τ be the solution returned by this algorithm. For every clause C_j we define a random variable Y_j which is 1 if clause C_j is satisfied, and 0 otherwise. Let W_j be the random variable which shows the contribution of the clause C_j to the total weight of the solution and let $W = \sum_{j=1}^{m} W_j = \sum_{j=1}^{m} w_j Y_j$. Then:

$$\mathrm{E}[W] = \sum_j w_j \mathrm{E}[Y_j]$$

$$
\begin{aligned}
&= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\
&= \sum_j w_j (1 - (\frac{1}{2})^{|C_j|}) \\
&\geq \frac{1}{2} \sum_j w_j \\
&\geq \frac{1}{2} OPT
\end{aligned}
$$

■

Note: if $|C_j| \geq k$ for all $1 \leq j \leq m$, then this is a $(1 - \frac{1}{2^k})$-approximation algorithm. This is a good algorithm if all the clauses are large. In particular, if all clauses have size 3 (Max-E3SAT) then this is a $\frac{7}{8}$-approximation. Surprisingly, for Max-E3SAT, this is the best possible approximation:

Theorem 8.3 (Hastad'97) *There is no α-approximation for Max-SAT (and in particular for Max-E3SAT) for $\alpha < \frac{7}{8}$, unless P = NP.*

Now we show how we can turn this randomized algorithm to a deterministic one using a general tool called the method of conditional probability.

8.1.2 De-randomization Using the Method of Conditional Probability

This is a general technique developed by Erdös and Spencer and can be used for many other problems. For this problem, we will use the following important property:

Lemma 8.4 *Suppose that we have assigned values $x_1 = a_1, \ldots, x_i = a_i$. Then we can compute the expected value of the solution in polynomial time.*

Proof: Let ϕ' be the reduced formula on variables $x_{i+1}, \ldots, x_n$ obtained from ϕ by substituting the values from $x_1 \ldots x_i$ and deleting the clauses that are already satisfied. Also, we remove variables $x_1, \ldots, x_i$ from every clause if the current assignment of that variable does not satisfy that clause. Clearly the expected value of solution for any random truth assignment to ϕ' can be computed in polynomial time (as in Theorem 8.2). Now we just add to this value the weights of the clauses of ϕ that were already satisfied by $x_i \ldots x_i$ we obtain the expected value of ϕ. ■

This suggests a simple algorithm.

• Consider the first variable x_1. It can be True or False. For each of True/False we can compute the expected value of the solution (assuming that all other variables' are assigned True/False uniformly randomly.

• If $\mathrm{E}[W|x_1 = T] > \mathrm{E}[W|x_1 = F]$, then we set $x_1 = T$, otherwise, we set $x_1 = F$.

• Let v be the value assigned to x_1 in the previous step.

$$
\begin{aligned}
\mathrm{E}[W] &= \mathrm{E}[W|x_1 = T] \cdot \Pr[x_1 = T] + \mathrm{E}[W|x_1 = F] \cdot \Pr[x_1 = F] \\
&= \frac{1}{2}(\mathrm{E}[W|x_1 = T] + E[W|x_1 = F])
\end{aligned}
$$

So if we set x_1 as above then:

$$
\mathrm{E}[W|x_1 = v] \geq \mathrm{E}[W] \geq \frac{1}{2} OPT
$$

• In general (by induction on i) if we have assigned the values for $x_1 \dots x_i$ then the expected value of the solution if $x_{i+1} = T$ and if $x_{i+1} = F$ can be computed in polynomial time by Lemma 8.4.

• As in for $i = 1$:

$$\begin{aligned} \mathrm{E}[W|x_1 = a_1, \dots, x_i = a_i] &= \mathrm{E}[W|x_1 = a_1, \dots, x_i = a_i, x_{i+1} = T] \cdot \Pr[x_{i+1} = T] \\ &\quad + \mathrm{E}[W|x_1 = a_1, \dots, x_i = a_i, x_{i+1} = F] \cdot \Pr[x_{i+1} = F] \end{aligned}$$

If we set x_{i+1} as said, then:

$$\begin{aligned} \mathrm{E}[W|x_1 = a_1, \dots, x_i = a_i, x_{i+1} = a_{i+1}] &\geq \mathrm{E}[W|x_1 = a_1, \dots, x_i = a_i] \\ \text{by induction} &\geq E[W] \\ &\geq \frac{1}{2} OPT \end{aligned}$$

We can use the same technique even if probability distance used is other than uniform.

Key points: We should be able to compute the conditional expected value for any possible outcome in polynomial-time.

Derandomized Alg1:
```
    for $i \leftarrow 1$ to $n$ do
        $W_T \leftarrow \mathrm{E}[W|x_1 = v_1, \dots, x_{i-1} = v_{i-1}, x_i = T]$;
        $W_F \leftarrow \mathrm{E}[W|x_1 = v_1, \dots, x_{i-1} = v_{i-1}, x_i = F]$;
        if $W_T > W_F$, then $v_i = T$;
        else $v_i \leftarrow F$;
```

According to the algorithm now we can compute:

$$\mathrm{E}[W|x_1 = a_1, \dots, x_i = v_i] = \sum_{j=1}^{m} w_j \cdot \mathrm{E}[Y_j|x_1 = a_1, \dots, x_i = v_i]$$

So we only need to compute $\mathrm{E}[Y_j||x_1 = a_1, \dots, x_i = v_i]$. We have:

$$\mathrm{E}[Y_j|x_1 = a_1, \dots, x_i = v_i] = \Pr[C_j \text{ satisfied}|x_1 = a_1, \dots, x_i = v_i]$$

Therefore, if one of $x_1 = a_1, \dots, x_i = v_i$ satisfies C_j then $\mathrm{E}[Y_j|x_1 = a_1, \dots, x_i = v_i] = 1$, otherwise $\mathrm{E}[Y_j|x_1 = a_1, \dots, x_i = v_i] = 1 - (\frac{1}{2})^k$, where k is the number of variables from $x_{i+1}, \dots, x_n$ in clause C_j.

8.1.3 A Better Algorithm Using Biased Coins

Now we introduce a better algorithm using biased coins. First, let's assume that all 1-clauses (i.e. clause of size 1) have non-negated variables. We set each $x_i = T$ with probability p ($\geq \frac{1}{2}$ to be defined). Then we return the truth assignment as the solution of the algorithm.

If C_j is a 1-clause, it is satisfied with prob p. If C_j is a $\geq$ 2-clause then let α be the number of negated variables in C_j, and β be the number of positive variables in C_j. So $\Pr[C_j \text{ is satisfied}] = 1 - p^\alpha \cdot (1-p)^\beta \geq 1 - p^{\alpha+\beta} \geq 1 - p^2$ (where we have used the fact $p \geq 1 - p$). This implies that:

Lemma 8.5 $\Pr[C_j$ *is satisfied*$] \geq \min[p, 1 - p^2]$.

If we set $p = 1 - p^2$ then $p = \frac{\sqrt{5}+1}{2} \simeq 0.618$. So with this value for p:

$$\mathrm{E}[W] = \sum_j w_j \Pr[C_j \text{ is satisfied}] \geq p \cdot \sum_j^m w_j \geq p \cdot OPT.$$

Therefore:

Theorem 8.6 *If all 1-clauses are non-negated then this is a p-approximation algorithm.*

What if there are some 1-clauses that have negated variables? For instance, if $C_j = \overline{x_i}$? One easy fix is to define a new variable $x'_i = \overline{x_i}$ and then replace $\overline{x_i}$ with x'_i and replace x_i with $\overline{x'_i}$. The only problem is when there are two 1-clauses, one containing x_i and one containing $\overline{x_i}$, let's say $C_j = x_i$ and $C_l = \overline{x_i}$. How can we handle that?

Without lose of generality, assume the $w_j \geq w_l$ (otherwise we first replace $\overline{x_i}$ with x'_i). Let's say $w'_i = w_l$ (i.e. w'_i is the weight of the smaller 1-clause). Let U be the set of indices of all clauses excluding those unit clauses that have negative literal (like $C_l above$), and V be the indices of the 1-clauses where a variable appears in negative forms in them. The key observation here is that the optimal solution cannot satisfy both C_j and C_l. So at best, we loose a weight equal to w'_i (which is w_l) from the lower bound of $\sum_j w_j$. That is:

$$OPT \leq \sum_j w_j - \sum_{i \in V} w'_i.$$

Therefore:

$$\begin{aligned}
\mathrm{E}[W] &= \sum_{j=1}^m w_j \cdot \Pr[C_j \text{ is satisfied}] \\
&\geq \sum_{j \in U} w_j \cdot \Pr[C_j \text{ is satisfied}] \\
&\geq p
\end{aligned}$$

$cdotsum_{j \in U} w_j$

$$\begin{aligned}
&\geq p \cdot \left[\sum_j w_j - \sum_{i \in V} w'_i\right] \\
&\geq p \cdot OPT
\end{aligned}$$

Therefore, the algorithm works even if we don't have the assumption on 1-clauses.

Theorem 8.7 *This is a p-approximation for Max-SAT with p defined above.*