6.891 Approximation Algorithms	February 18, 2000
Lecture 3	
Lecturer: David P. Williamson	*Scribe: David G. Andersen*

1 Randomization: MAX SAT

We now turn to uses of randomization in developing approximation algorithms. For this, we need to define randomized approximation algorithms.

Definition 1 *We have a randomized α-approximation algorithm if the algorithm runs in randomized polynomial time, and if it produces a solution that is always within a factor of α of an optimal solution*

- *with high probability (that is, at least $1 - \frac{1}{n^c}$ for some constant $c > 1$)*
- *OR in expectation,*

over the random choices of the algorithm.

Randomized polynomial time is polynomial time extended so that one `random()` call takes a single unit of time. We will be slightly vague about what we want our `random()` primitive to do; most of the time it will be sufficient to assume that it returns a random number from [0,1].

We begin by introducing the maximum satisfiability problem (MAX SAT) and a "dumb" randomized algorithm for it.

MAX SAT

- **Input:**
  - n boolean variables $x_1, x_2, \ldots, x_n$
  - m clauses $C_1, C_2, \ldots, C_m$ (e.g. $x_3 \vee \bar{x}_5 \vee \bar{x}_7 \vee x_{11}$)
  - weight $w_i \geq 0$ for each clause C_i
- **Goal:** Find an assignment of TRUE/FALSE for the x_i that maximizes total weight of satisfied clauses. (e.g. $x_3 \vee \bar{x}_5 \vee \bar{x}_7 \vee x_{11}$ is satisfied if x_3 is set TRUE, x_5 is set FALSE, x_7 is set FALSE, or x_{11} is set TRUE).

1.1 Flipping coins (Johnson's algorithm)

What is the dumbest possible use of randomization for this problem? The dumbest possible thing is to set each variable to be true randomly and independently; i.e. set x_i to true with probability $1/2$ independently.

Theorem 1 *($\approx$ Johnson '74) This randomized algorithm is a $\frac{1}{2}$-approximation algorithm.*

Proof: Consider a random variable Y_j such that

$$Y_j = \begin{cases} 1 & \text{if clause } j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$W = \sum_{j=1}^{m} w_j Y_j.$$

Then, by linearity of expectation

$$\begin{aligned} E[W] &= \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[\text{clause } j \text{ is satisfied}] \\ &= \sum_{j=1}^{m} w_j \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT, \end{aligned}$$

where $l_j = \#$ literals in clause j, since $l_j \geq 1$ and the sum of the weights of all clauses is an upper bound on the value of an optimal solution. $\square$

Observe that if $l_j \geq k\ \forall j$, then we have a $(1 - (\frac{1}{2})^k)-$approximation algorithm. Thus Johnson's algorithm is bad when clauses are short, but good if clauses are long.

Although this seems like a pretty naive algorithm, a recent theorem shows that in fact this is the best that can be done in some cases. MAX E3SAT is the subset of MAX SAT instances in which each clause has exactly three literals in it. Note that Johnson's algorithm gives a $\frac{7}{8}$-approximation algorithm in this case.

Theorem 2 *(Håstad '97) If* MAX E3SAT *has an α-approximation algorithm, $\alpha > \frac{7}{8}$, then $P = NP$.*

1.2 Derandomization

We can make the algorithm deterministic using the Method of Conditional Expectations (Spencer, Erdös). This method is very general, and allows for the derandomization of many randomized algorithms.

In the simple case, with the same set of boolean variables $x_1, x_2, \ldots x_n$, set $x_2, \ldots, x_n$ randomly, but set x_1 in the "best" way:

- Assume x_1 both TRUE and FALSE, and find the expected value of the solution:

$$\begin{array}{rc} if & E[W|x_1 \leftarrow TRUE] > E[W|x_1 \leftarrow FALSE] \\ then & x_1 \leftarrow TRUE \\ else & x_1 \leftarrow FALSE \end{array}$$

By the definition of conditional expectations, we can express $E[W]$ in terms of these two conditional expectations:

$$\begin{aligned} E[W] &= E[W|x_1 \leftarrow TRUE] \cdot Pr[x_1 \leftarrow TRUE] \\ &\quad + E[W|x_1 \leftarrow FALSE] \cdot Pr[x_1 \leftarrow FALSE] \\ &= \frac{1}{2}(E[W|x_1 \leftarrow True] + E[W|x_1 \leftarrow FALSE]) \end{aligned}$$

If x_1 is set to b_1 as above, then the overall expectation is:

$$E[W|x_1 \leftarrow b_1] \geq E[W] \geq \frac{1}{2} \cdot OPT$$

So we don't lose anything by setting x_1 in this manner. Note that this does not depend on $P = \frac{1}{2}$ – we can generalize randomization to other coin flip probabilities later.

In general, suppose that we've already set $x_1, x_2, \ldots, x_i$. How do we set x_{i+1}?

$$\begin{array}{rc} if & E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow TRUE] \geq \\ & E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow FALSE] \\ then & set\ x_{i+1} \leftarrow TRUE \\ else & set\ x_{i+1} \leftarrow FALSE \end{array}$$

The math works out just like before:

$$\begin{aligned} & E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i] = \\ & \qquad E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow TRUE] \cdot Pr[x_{i+1} \leftarrow TRUE] \\ & \quad + \ E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow FALSE] \cdot Pr[x_{i+1} \leftarrow FALSE] \end{aligned}$$

Setting x_{i+1} as above yields

$$\begin{aligned} E[W|x_1 \leftarrow b_w, \ldots, x_{i+1} \leftarrow b_{i+1}] &\geq E[W|x_1 \leftarrow b_w, \ldots, x_i \leftarrow b_i] \\ &\geq E[W] \\ &\geq \frac{1}{2} \cdot OPT \end{aligned}$$

Observe that of course $E[W|x_1 \leftarrow b_1, \ldots, x_n \leftarrow b_n]$ is the value of the deterministically determined solution to the problem, so that by induction, the value of this solution is at least $\frac{1}{2} \cdot OPT$.

This results in the following algorithm for derandomization:

Derandomized Dumb

> For $i \leftarrow 1$ to n
> $W_T \leftarrow E[W|x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_{i-1} \leftarrow b_{i-1}, x_i \leftarrow TRUE]$
> $W_F \leftarrow E[W|x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_{i-1} \leftarrow b_{i-1}, x_i \leftarrow FALSE]$
>   If $W_T \geq W_F$
> $x_i \leftarrow$ TRUE
>   else
> $x_i \leftarrow$ FALSE.

How do we calculate $E[W|x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i]$ in this algorithm? By linearity of expectations, we know that

$$E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i] = \sum_j w_j E[X_j|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i].$$

Furthermore, we know that

$$E[X_j|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i] = \Pr[\text{clause } j \text{ is satisfied } |x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i].$$

It is not hard to determine that

$$\begin{aligned} &\Pr[\text{clause } j \text{ is satisfied } |x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_1] \\ &= \begin{cases} 1 & \text{if } x_1, \ldots, x_i \text{ already satisfy clause } j \\ 1 - (\frac{1}{2})^k & \text{otherwise when } k\text{=\# variables of } x_{i+1}, \ldots, x_n \text{ in clause } j \end{cases} \end{aligned}$$

Consider, for example, the clause $x_3 \vee \bar{x}_5 \vee \bar{x}_7 \vee x_{11}$. It is not hard to see that

$$\Pr[\text{clause satisfied } |x_1 \leftarrow T, x_2 \leftarrow F, x_3 \leftarrow T, x_4 \leftarrow F] = 1,$$

since $x_3 \leftarrow T$ satisfies the clause. On the other hand,

$$\Pr[\text{clause satisfied } |x_1 \leftarrow T, x_2 \leftarrow F, x_3 \leftarrow F, x_4 \leftarrow F] = 1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8},$$

since only the "bad" settings of x_5, x_7, and x_{11} will make the clause unsatisfied.

We can think of this sort of derandomization as a walk down a tree, with each choice of a variable value corresponds to a choice of a branch of the tree. The expectation at each node is the average of the expected values of the nodes below it. Thus we can assure ourselves that as we make these choices, our expected value is staying as least as large as it was initially. Thus the factor of $\frac{1}{2}$ still holds in this derandomized case.

The Method of Conditional Expectations allows us to give deterministic variants of randomized algorithms for most of the randomized algorithms we discuss. Why discuss the randomized algorithms, then? It turns out that usually the randomized algorithm is easier to state and analyze that its corresponding deterministic variant.

1.3 Flipping Bent Coins

As a stepping stone to better approximation algorithms for the maximum satisfiability problem, we consider what happens if we bias the probabilities for each boolean variable. To do this, we restrict our attention for the moment to MAX SAT instances in which all length 1 clauses are not negated, and we set x_i true with probability p (at least 1/2, to be determined later) independently.

Lemma 3 $\Pr[\textit{clause } j \textit{ is satisfied}] \geq \min(p, 1-p^2)$

Proof: If $l_j = 1$ then

$$\Pr[C_j \text{ is satisfied}] = p,$$

since every length 1 clause appears positively. If $l_j \geq 2$ then

$$\Pr[C_j \text{ is satisfied}] = 1 - p^a(1-p)^b \geq 1 - p^{a+b} \geq 1 - p^2,$$

where a is the number of negated literals in the clause and b is the number of unnegated literals. The first inequality follows since $p \geq (1-p)$, and the second since $a + b = l_j \geq 2$. □

We set $p = 1 - p^2 \Rightarrow p = \frac{1}{2}(\sqrt{5} - 1) \approx 0.618$.

Theorem 4 *(Lieberherr, Specker '81) Flipping biased coins as above is a p-approximation algorithm for MAX SAT when all length 1 clauses are not negated.*

Proof:

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

□

We can actually extend this result to the general case when the length 1 clauses can be negated or non-negated. First note that if only a variable's negation appears as a length 1 clause, then we can redefine our variables to have a non-negated variable in that clause instead. The only case we really have to worry about, then, is the case where both a variable and its negation appear as length 1 clauses. In this case we can get the result above by making a stronger statement about the optimal value, OPT.

WLOG, assume that the weight of clause x_i is greater than weight of clause $\bar{x}_i$. Let v_i = wt. of $\bar{x}_i$. This allows a better bound on OPT:

$$OPT \leq \sum_j w_j - \sum_i v_i,$$

since the optimal solution can only satisfy either x_i or $\bar{x}_i$.

Theorem 5 *(Lieberherr, Specker '81) Flipping biased coins is a p-approximation algorithm for MAX SAT.*

Proof: Let U denote the set of indices of clauses excluding unit clauses $\bar{x}_i$.Let C_j denote the j^{th} clause. Then we have

$$\begin{aligned} E[W] = \sum_j w_j Pr[\text{clause j satisfied}] &\geq \sum_{j \in U} w_j Pr[C_j \text{ satisfied}] \\ &\geq p \cdot \sum_{j \in U} w_j \\ &\geq p \cdot [\sum_j w_j - \sum_i v_i] \\ &\geq p \cdot OPT \end{aligned}$$

$\square$

1.4 Randomized Rounding

We now consider what would happen if we tried to give different biases to determine each x_i. To do that, we turn to using a linear programming relaxation of the problem. We model MAX SAT as the following integer program, in which we introduce a variable z_j for every clause and a variable y_i for each boolean variable x_i.

$$\begin{aligned} z_j &= \begin{cases} 1 & \text{if clause } j \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases} \\ y_i &= \begin{cases} 1 & \text{if } x_i \leftarrow \text{true} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

We let P_j be the indices of the variables that occur positively in clause j and N_j be the indices of the variables that occur negatively in clause j.

$$
\begin{aligned}
\text{Max} \quad & \sum_j w_j z_j \\
\text{subject to:} \quad & \\
& \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \qquad\qquad \forall C_j : \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i \\
& y_i \in \{0,1\} \\
& 0 \leq z_j \leq 1.
\end{aligned}
$$

This IP models MAX-SAT exactly, but is, of course, not polynomial-time solvable (unless $P = NP$).

To obtain an LP, we relax $y_i \in \{0,1\}$ to $0 \leq y_i \leq 1$. Note that if z_{LP} is the LP optimum and OPT is the integral optimum, then $z_{LP} \geq OPT$.

We now consider a very general technique introduced by Raghavan and Thompson called *randomized rounding*. Given a 0-1 integer program relaxed to an linear program, they suggest creating a 0-1 solution $\hat{v}$ from the linear solution v^* by setting $\hat{v}_i$ to 1 with probability v_i^*. Here we do this by setting x_i to true with probability y_i^* independently.

Theorem 6 *(Goemans, W '94) Randomized rounding is a $(1 - \frac{1}{e})$-approximation algorithm, where $1 - \frac{1}{e} \approx 0.632$.*

Proof: We need two facts to prove this theorem.

Fact 1 *For any nonegative $a_1, \ldots, a_k$, the geometric mean is $\leq$ the arithmetic mean.*

$$\sqrt[k]{a_1 a_2 \ldots a_k} \leq \frac{1}{k}(a_1 + a_2 + \cdots + a_k)$$

Fact 2 *If $f(x)$ is concave on $[l,u]$ (that is, $f''(x) \leq 0$ on $[l,u]$), and $f(l) \geq al + b$ and $f(u) \geq au + b$, then the funcion is lower-bounded by a line through the endpoints at $[l,u]$*

$$f(x) \geq ax + b \quad on\ [l,u].$$

Consider an arbitrary clause C_j.

$$
\begin{aligned}
\Pr[\text{clause is not satisfied}] &= \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
&\leq \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j} && (1) \\
&= \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right] && (2) \\
&\leq \left(1 - \frac{z_j^*}{l_j} \right)^{l_j}, && (3) \\
& && (4)
\end{aligned}
$$

so that

$$\Pr[\text{clause satisfied}] \geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*, \tag{5}$$

where (1) follows from Fact 1, (3) follows by the LP constraint, and (5) follows by Fact 2, since

$$\begin{aligned} z_j^* = 0 \ &\Rightarrow \ 1 - (1 - z_j^*/l_j)^{l_j} = 0 \\ z_j^* = 1 \ &\Rightarrow \ 1 - (1 - z_j^*/l_j)^{l_j} = 1 - \left(1 - \frac{1}{l_j}\right)^{l_j} \end{aligned}$$

and $1 - (1 - z_j^*/l_j)^{l_j}$ is concave.

Therefore,

$$\begin{aligned} E[W] &= \sum_j w_j \Pr[\text{clause } j \text{ is satisfied}] \\ &\geq \min_k \left[1 - \left(1 - \frac{1}{k}\right)^k\right] \sum_j w_j z_j^* \\ &\geq \min_k \left[1 - \left(1 - \frac{1}{k}\right)^k\right] \cdot OPT \geq \left(1 - \frac{1}{e}\right) \cdot OPT, \end{aligned}$$

since $(1 - \frac{1}{x})^x$ converges to e^{-1} from below. □

Observe that this algorithm does well when all clauses are short. If $l_j \leq k$ for all j, then the performance guarantee becomes $1 - (1 - 1/k)^k$.

1.5 A Best-of-Two algorithm

In the previous section we used the technique of randomized rounding to improve a .618-approximation algorithm to a .632-approximation algorithm for MAX SAT. This doesn't seem like much of an improvement.

But notice that Johnson's algorithm and the randomized rounding algorithm have conflicting bad cases. Johnson's algorithm is bad when clauses are short, whereas randomized rounding is bad when clauses are long. It turns out we can get an approximation algorithm that is much better than either algorithm just by taking the best solution of the two produced by the two algorithms.

Best-of-two

```
Run Johnson's algorithm, get assign x^1 of weight W_1
Run randomized rounding, get assign x^2 of weight W_2
If W_1 ≥ W_2
    return x^1
else
    return x^2.
```

Theorem 7 *(Goemans, W'94)* Best-of-two *is a $\frac{3}{4}$-approximation algorithm for MAX SAT.*

Proof:

$$\begin{aligned}
E[\max(W_1, W_2)] &\geq E\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] \\
&= \sum_j w_j \left(\frac{1}{2}\Pr[\text{clause } j \text{ is satisfied by DumbRandom}]\right. \\
&\qquad\qquad \left. + \frac{1}{2}\Pr[\text{clause } j \text{ is satisfied by RandomRound}]\right) \\
&\geq \sum_j w_j \left[\frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^{l_j}\right) + \frac{1}{2}\left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*\right] \\
&\geq \sum_j w_j \left[\frac{3}{4} z_j^*\right] \qquad (6) \\
&= \frac{3}{4}\sum_j w_j z_j^* \\
&\geq \frac{3}{4} OPT.
\end{aligned}$$

We need to prove inequality (6), which follows if

$$\frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^{l_j}\right) + \frac{1}{2}\left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^* \geq \frac{3}{4} z_j^*.$$

The cases $l_j = 1, 2$ are easy:

$$\begin{aligned}
l_j = 1 \quad &\Rightarrow \quad \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}z_j^* \geq \frac{3}{4}z_j^* \\
l_j = 2 \quad &\Rightarrow \quad \frac{1}{2}\cdot\frac{3}{4} + \frac{1}{2}\cdot\frac{3}{4}z_j^* \geq \frac{3}{4}z_j^*
\end{aligned}$$

For the case $l_j \geq 3$, we take the minimum possible value of the two terms:

$$l_j \geq 3 \quad \Rightarrow \quad \frac{1}{2}\cdot\frac{7}{8} + \frac{1}{2}\left(1 - \frac{1}{e}\right) z_j^* \geq \frac{3}{4} z_j^*$$

□

1.6 Non-linear Randomized Rounding

Nothing says that we had to take the output of the linear program directly into our randomized rounding scheme. We could have just as easily used some function of the output to generate rounding probabilities. This will result in a so-called non-linear randomized rounding approach. As an example, suppose we pick any function $g(y)$ such that $1-4^{-y} \leq g(y) \leq 4^{y-1}$ for $y \in [0,1]$. Then set each x_i true with probability $g(y_i^*)$ independently.

Theorem 8 *(Goemans, W '94) Nonlinear-Round is a 3/4-approximation algorithm for MAX SAT.*

Proof: Recall **Fact 2** in previous section: If $f(x)$ is concave in $[l,u]$, and $f(l) \geq al+b, f(u) \geq au+b$, then $f(x) \geq ax+b$ on $[l,u]$.

Then

$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j}(1-g(y_i^*)) \prod_{i \in N_j} g(y_i^*) \\
&\leq \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^*-1} \\
&= 4^{-\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j}(1-y_i^*)\right)} \\
&\leq 4^{-z_j^*},
\end{aligned}$$

where the last inequality follows from the corresponding LP constraint. Thus

$$\Pr[C_j \text{ satisfied}] \geq 1-4^{-z_j^*} \geq \frac{3}{4} z_j^*,$$

the second inequality follows from Fact 2, since the function is concave with respect to z; $f(0)=0, f(1)=\frac{3}{4}$.

Hence,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}] \geq \frac{3}{4} \sum_j w_j z_j^* \geq \frac{3}{4} OPT$$

$\square$

Note that other functions will also work for $\frac{3}{4}$ approximations.

So where do you go from here? It turns out that this is the end of the line insofar as comparing against the linear programming bound. To see this, consider the instance $x_1 \vee x_2$,$x_1 \vee \bar{x}_2$, $\bar{x}_1 \vee x_2$, $\bar{x}_1 \vee \bar{x}_2$, where each clause has weight 1. An optimal LP solution for this instance sets $y_i = \frac{1}{2}$ for all i and $z_j = 1$ for all clauses (indeed, this is true for any instance which has no length 1 clauses). Thus $Z_{LP} = 4$, and the best bound we can get comparing our solution against Z_{LP} is $\frac{3}{4}$. Observe that in the case there are no length 1 clauses, the optimal solution of $y_i = \frac{1}{2}$ for all i gives no information about how to set the variables; essentially we are back to Johnson's algorithm in this case!

The best known approximation algorithm for MAX SAT so far is ≈ 0.7846-approximation algorithm using semidefinite programming. If a certain conjecture turns out to be true, we can achieve is .8331. The known limit is $\frac{7}{8}$ overall, from MAX E3SAT.

Research question: Can you get a $\frac{3}{4}$-approximation algorithm for MAX SAT without solving an LP?