5 Random sampling and randomized rounding of LPs

- Sometimes it turns out to be useful to allow our algorithms to make random choices
- I.e., the algorithm can flip a coin, or flip a biased coin, or draw a value uniformly from a given interval
- The performance guarantee of such an approximation algorithm is then the expected value of the solution produced relative to the that of an optimal solution, where the expectation is taken over the random choices of the algorithm

- We are often able to show that randomized approximation algorithms can be derandomized
- We can the method of conditional expectations to produce a deterministic version of the algorithm that has the same performance guarantee as the randomized version
- It is often much simpler to state and analyze the randomized version of the algorithm than the deterministic version from derandomization
- Thus randomization gains us simplicity in our algorithm design and analysis, while derandomization ensures that the performance guarantee can be obtained deterministically
5.1 Simple algorithm for MAX SAT

- In MAX SAT, the input consists of \( n \) Boolean variables \( x_1, \ldots, x_n \), \( m \) clauses \( C_1, \ldots, C_m \) (e.g., \( x_3 \lor \overline{x}_5 \lor x_{11} \)), and a nonnegative weight \( w_j \) for each clause \( C_j \).
- The objective of the problem is to find an assignment of true/false to the \( x_i \) that maximizes the weight of the satisfied clauses.
- A clause is satisfied if one of the unnegated variables is set to true, or one of the negated variables is set to false.

- A variable \( x_i \) or a negated variable \( \overline{x}_i \) is a *literal*.
- Each clause consists of some number of literals.
- A variable \( x_i \) is called a positive literal and a negated variable \( \overline{x}_i \) is called a negative literal.
- The number of literals in a clause is its *length*.
- We denote the length of a clause \( C_j \) by \( l_j \).
- Clauses of length one are called unit clauses.
- We assume that no literal is repeated in a clause and that at most one of \( x_i \) and \( \overline{x}_i \) appears in one clause.
- It is natural to assume that the clauses are distinct, since we can simply sum the weights of two identical clauses.
A very straightforward use of randomization for MAX SAT is to set each $x_i$ to true independently with probability $1/2$.

An alternate perspective on this algorithm is that we choose a setting of the variables uniformly at random from the space of all possible settings.

**Theorem 5.1:** Setting each $x_i$ to true with probability $1/2$ independently gives a randomized $1/2$-approximation algorithm for the maximum satisfiability problem.

**Proof:** Consider a random variable (RV) $Y_j$ that is 1 if clause $j$ is satisfied and 0 otherwise. Let $W$ be a RV that is equal to the total weight of the satisfied clauses, so that $W = \sum_{j=1}^{m} w_j Y_j$. Let OPT denote the optimum value of the MAX SAT instance. Then, by linearity of expectation, and the definition of the expectation of a 0-1 RV, we know that

$$E[W] = \sum_{j=1}^{m} w_j E[Y_j]$$

$$= \sum_{j=1}^{m} w_j \Pr[\text{clause } C_j \text{ satisfied}]$$
For each clause $c_j$, the Pr that it is not satisfied is the Pr that each pos. literal in $c_j$ is set to false and each neg. literal in $c_j$ is set to true, each of which happens with probability $1/2$ independently; hence

$$\Pr[\text{clause } c_j \text{ satisfied}] = \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}.$$ 

The last inequality follows from the fact that $l_j \geq 1$.

$$E[W] \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2} \text{OPT}.$$ 

The last inequality follows from the fact that the total weight is an upper bound on $\text{OPT}$, since each weight is assumed to be nonnegative.

- Observe that if $l_j \geq k$ for each clause $j$, then the algorithm is a $(1 - (1/2)^k)$-approximation algorithm for such instances.
- The performance of the algorithm is better on MAX SAT instances consisting of long clauses.
- Although this seems like a pretty naive algorithm, a hardness theorem shows that this is the best that can be done in some cases.
- Consider the case in which $l_j = 3$ for all clauses $j$; this restriction of the problem is sometimes called MAX E3SAT, since there are exactly 3 literals in each clause.
The analysis above shows that the randomized algorithm gives an approximation algorithm with performance guarantee \((1 - (1/2)^3) = 7/8\).

A truly remarkable result shows that nothing better is possible for these instances unless \(P = NP\).

**Theorem 5.2**: If there is an \((\frac{7}{8} + \epsilon)\)-approximation algorithm for MAX E3SAT for any constant \(\epsilon > 0\), then \(P = NP\).

### 5.2 Derandomization

- We now show how the algorithm of the preceding section for the maximum satisfiability problem can be derandomized by replacing the randomized decision of whether to set \(x_i\) to true with a deterministic one that will preserve the expected value of the solution.
- These decisions will be made sequentially: the value of \(x_1\) is determined first, then \(x_2\), and so on.
- How should \(x_1\) be set so as to preserve the expected value of the algorithm?
• Assume for now that we will make the choice of $x_1$ deterministically, and all other variables will be set true with probability $1/2$ as before.

• Then the best way to set $x_1$ is that which will maximize the expected value of the resulting solution:
  - We should determine the expected value of $W$, the weight of satisfied clauses, given that $x_1$ is set to true, and that given that $x_1$ is set to false.
  - Then we set $x_1$ to whichever value maximizes the expected value of $W$.

It makes intuitive sense that this should work, since the maximum is always greater than an average, and the expected value of $W$ is the average of its expected value given the two possible settings of $x_1$.

In this way, we maintain an algorithmic invariant that the expected value is at least half the optimum, while having fewer random variables left.

More formally, if $E[W \mid x_1 \leftarrow \text{true}] \geq E[W \mid x_1 \leftarrow \text{false}]$, then we set $x_1$ true, otherwise we set it to false.
• By the definition of conditional expectations
  \[ E[W] = E[W \mid x_1 \leftarrow \text{true}] \Pr[x_1 \leftarrow \text{true}] + \]
  \[ E[W \mid x_1 \leftarrow \text{false}] \Pr[x_1 \leftarrow \text{false}] \]
  \[ = \frac{1}{2} (E[W \mid x_1 \leftarrow \text{true}] + E[W \mid x_1 \leftarrow \text{false}]), \]

• if we set \( x_1 \) to truth value \( b_1 \) so as to maximize the conditional expectation, then
  \[ E[W \mid x_1 \leftarrow b_1] \geq E[W] \]

• The deterministic choice of how to set \( x_1 \) guarantees an expected value no less than the expected value of the completely randomized algorithm.

• Assuming for the moment that we can compute these conditional expectations, the deterministic decision of how to set the remaining variables is similar

• Assume that we have set variables \( x_1, \ldots, x_i \) to truth values \( b_1, \ldots, b_i \) respectively

• How shall we set variable \( x_{i+1} \)?

• Again, assume that the remaining variables are set randomly

• Then the best way to set \( x_{i+1} \) is so as to maximize the expected value given the previous settings of \( x_1, \ldots, x_i \)
• So if \( E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{true}] \geq E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{false}] \), we set \( x_{i+1} \) to true (thus \( b_{i+1} \leftarrow \text{true} \)), otherwise we set \( x_{i+1} \) to false (\( b_{i+1} \leftarrow \text{false} \)).

• Then since 
\[
E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i] = E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{true}] \Pr [x_{i+1} \leftarrow \text{true}] + E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{false}] \Pr [x_{i+1} \leftarrow \text{false}]
\]
\[
= \frac{1}{2} (E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{true}] + E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow \text{false}])
\]
setting \( x_{i+1} \) to truth value \( b_{i+1} \) ensures that

\[
E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow b_{i+1}] \geq E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]
\]

• By induction, this implies that 
\[
E[W | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow b_{i+1}] \geq E[W]
\]

• We continue this process until all \( n \) variables have been set.

• Since the conditional expectation given by the setting all variables, \( E[W | x_1 \leftarrow b_1, \ldots, x_n \leftarrow b_n] \), is simply the value of the solution given by the deterministic algorithm, we know that the value of the solution returned is at least
\[
E[W] \geq \frac{1}{2} \text{OPT}
\]
These conditional expectations are not difficult to compute.

By definition,

$$E[W \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$$

$$= \sum_{j=1}^{m} w_j E[Y_j \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$$

$$= \sum_{j=1}^{m} w_j \Pr[\text{Clause } C_j \text{ satisfied} \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$$

The probability that clause $C_j$ is satisfied given that $x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i$ is easily seen to be 1 if the settings of $x_1, \ldots, x_i$ already satisfy the clause.

Otherwise the probability is $1 - (1/2)^k$, where $k$ is the number of literals in the clause that remain unset by this procedure.

For example, consider the clause $x_3 \lor \bar{x}_3 \lor \bar{x}_7$.

It is the case that

$$\Pr[\text{Clause satisfied} \mid x_1 \leftarrow \text{true}, x_2 \leftarrow \text{false}, x_3 \leftarrow \text{true}] = 1$$

since setting $x_3$ to true satisfies the clause.

On the other hand,

$$\Pr[\text{Clause satisfied} \mid x_1 \leftarrow \text{true}, x_2 \leftarrow \text{false}, x_3 \leftarrow \text{false}]$$

$$= 1 - \left(\frac{1}{2}\right)^2 = \frac{3}{4},$$

since the clause will be unsatisfied only if $x_5$ and $x_7$ are set true, an event that occurs with prob. $1/4$. 
5.3 Flipping biased coins

- Biasing the probability with which we set $x_i$ is actually helpful; i.e., we will set $x_i$ true with some probability not equal to $1/2$
- To do this, it is easiest to start by considering only MAX SAT instances with no unit clauses $\neg x_i$, that is, no negated unit clauses
- We can remove this assumption
- Suppose now we set each $x_i$ to be true independently with probability $p > 1/2$

Lemma 5.4: If each $x_i$ is set to true with probability $p > 1/2$ independently, then the probability that any given clause is satisfied is at least $\min(p, 1 - p^2)$ for MAX SAT instances with no negated unit clauses.

Proof: If the clause is a unit clause, then the probability the clause is satisfied is $p$, since it must be of the form $x_i$, and the probability $x_i$ is set true is $p$. If the clause has length at least two, then the probability that the clause is satisfied is $1 - p^a(1 - p)^b$, where $a$ is the number of negated variables in the clause and $b$ is the number of unnegated variables in the clause, so that $a + b = l_j \geq 2$. Since $p > \frac{1}{2} > 1 - p$, this probability is at least $1 - p^{a+b} = 1 - p^{l_j} \geq 1 - p^2$, and the lemma is proved. ■
• We can obtain the best performance guarantee by setting $p = 1 - p^2$
• This yields $p = \frac{1}{2} (\sqrt{5} - 1) \approx 0.618$

**Theorem 5.5:** Setting each $x_i$ to true with probability $p$ independently gives a randomized $\min(p, 1 - p^2)$-approximation algorithm for MAX SAT instances with no negated unit clauses.

**Proof:** This follows since

$$E[W] = \sum_{j=1}^{m} w_j \Pr[\text{Clause } C_j \text{ satisfied}]$$

$$\geq \min(p, 1 - p^2) \sum_{j=1}^{m} w_j \geq \min(p, 1 - p^2) \text{OPT}.$$ 

We would like to extend this result to all MAX SAT instances.

To do this, we will use a better bound on OPT than $\sum_{j=1}^{m} w_j$

Assume that for every $i$ the weight of the unit clause $x_i$ appearing in the instance is at least the weight of the unit clause $\overline{x}_i$; this is w.l.o.g. since we could negate all occurrences of $x_i$ if the assumption is not true.

Let $v_i$ be the weight of the unit clause $\overline{x}_i$ if it exists in the instance, and let $v_i$ be zero otherwise.
Lemma 5.6: \( \text{OPT} \leq \sum_{j=1}^{m} w_j - \sum_{i=1}^{n} v_i \).

Proof: For each \( i \), the optimal solution can satisfy exactly one of \( x_i \) and \( \bar{x}_i \). Thus the weight of the optimal solution cannot include both the weight of the clause \( x_i \) and the clause \( \bar{x}_i \). Since \( v_i \) is the smaller of these two weights, the lemma follows. ■

- We can now extend the result

**Theorem 5.7:** We can obtain a randomized \( \frac{1}{2} (\sqrt{5} - 1) \)-approximation algorithm for MAX SAT.

- This algorithm can be derandomized using the method of conditional expectations

5.4 Randomized rounding

- We can do still better by giving each variable its own bias
- Recall that in randomized rounding, we first set up an integer programming formulation of the problem at hand in which there are 0-1 integer variables
- In this case we will create an IP with a 0-1 variable \( y_i \) for each Boolean variable \( x_i \) such that \( y_i = 1 \) corresponds to \( x_i \) set true
The IP is relaxed to a linear program by replacing the constraints $y_i \in \{0,1\}$ with $0 \leq y_i \leq 1$, and the LP relaxation is solved in polynomial time.

The central idea of randomized rounding is that the fractional value $y_i^*$ is interpreted as the probability that $y_i$ should be set to 1.

In this case, we set each $x_i$ to true with probability $y_i^*$ independently.

In addition to the variables $y_i$, we introduce a variable $z_j$ for each clause $C_j$ that will be 1 if the clause is satisfied and 0 otherwise.

For each clause $C_j$ let $P_j$ be the indices of the variables $x_i$ that occur positively in the clause, and let $N_j$ be those that are negated in it.

We denote the clause $C_j$ by

$$\bigvee_{i \in P_j} x_i \bigvee_{i \in N_j} \bar{x}_i$$

Then the inequality

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j$$

must hold for clause $C_j$. 
Each variable that occurs positively in the clause is set to false (and its corresponding $y_i \leftarrow 0$) and each variable that occurs negatively is set to true (and its corresponding $y_i \leftarrow 1$).

Then the clause isn’t satisfied, and $z_j$ must be 0.

This inequality yields the following IP formulation of the MAX SAT problem:

$$\text{maximize } \sum_{j=1}^{m} w_j z_j$$

subject to

$$\sum_{i \in P} y_i + \sum_{i \in N} (1 - y_i) \geq z_j, \quad \forall c_j = \bigvee_{i \in P} x_i \lor \bigvee_{i \in N} \bar{x}_i, \quad 0 \leq y_i \leq 1, \quad i = 1, \ldots, n$$

$$0 \leq z_j \leq 1, \quad j = 1, \ldots, m$$

If $Z^*_{IP}$ is the optimal value of this IP, then it is not hard to see that $Z^*_{IP} = OPT$.

The corresponding LP relaxation of this IP is

$$\text{maximize } \sum_{j=1}^{m} w_j z_j$$

subject to

$$\sum_{i \in P} y_i + \sum_{i \in N} (1 - y_i) \geq z_j, \quad \forall c_j = \bigvee_{i \in P} x_i \lor \bigvee_{i \in N} \bar{x}_i, \quad 0 \leq y_i \leq 1, \quad i = 1, \ldots, n$$

$$0 \leq z_j \leq 1, \quad j = 1, \ldots, m$$

If $Z^*_{LP}$ is the optimal value of this LP, then clearly $Z^*_{LP} \geq Z^*_{IP} = OPT$. 
• Let \((y^*, z^*)\) be an optimal solution to the LP relaxation

• Consider now the result of using randomized rounding, and setting \(x_i\) to true with probability \(y_i^*\) independently

• The following compares the arithmetic and geometric means of a set of numbers

**Fact 5.8** (Arithmetic-geometric mean inequality): For any nonnegative \(a_1, \ldots, a_k\),

\[
\left( \prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \sum_{i=1}^{k} a_i.
\]

**Fact 5.9**: If a function \(f(x)\) is concave on the interval \([0,1]\) (that is, \(f''(x) \leq 0\) on \([0,1]\)), and \(f(0) = a\) and \(f(1) = b + a\), then \(f(x) \geq bx + a\) for \(x \in [0,1]\).
**Theorem 5.10:** Randomized rounding gives a \((1 - \frac{1}{e})\)-approximation algorithm for MAX SAT.

**Proof:** As in the analyses of the algorithms in the previous sections, the main difficulty is analyzing the probability that a given clause \(C_j\) is satisfied. Pick an arbitrary clause \(C_j\). Then, by applying the arithmetic-geometric mean inequality, we see that

\[
\Pr[\text{clause } C_j \text{ not satisfied}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
\leq \left[ \frac{1}{l_j} \left( \sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}
\]

By rearranging terms, we can derive that

\[
\left[ \frac{1}{l_j} \left( \sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j} = \left[ 1 - \frac{1}{l_j} \left( \sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right]^{l_j}
\]

By invoking the corresponding inequality from the LP,

\[
\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \geq z_j^*
\]

we see that

\[
\Pr[\text{clause } C_j \text{ not satisfied}] \leq \left( 1 - \frac{z_j^*}{l_j} \right)^{l_j}
\]

The function \(f(z_j^*) = 1 - \left( 1 - \frac{z_j^*}{l_j} \right)^{l_j}\) is concave for \(l_j \geq 1\).
Then by using Fact 5.9,
\[
\Pr[\text{clause } C_j \text{ satisfied}] \geq 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{t_j} \geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{t_j}\right]z_j^*
\]
Therefore the expected value of the randomized rounding algorithm is
\[
E[W] = \sum_{j=1}^{m} w_j \Pr[\text{clause } C_j \text{ satisfied}]
\geq \sum_{j=1}^{m} w_j z_j^* \left[1 - \left(1 - \frac{1}{l_j}\right)^{t_j}\right]
\geq \min_{k \geq 1} \left[1 - \left(1 - \frac{1}{k}\right)^k\right] \sum_{j=1}^{m} w_j z_j^*
\]

Note that \(1 - \left(1 - \frac{1}{k}\right)^k\) is a nonincreasing function in \(k\) and that it approaches \(1 - \frac{1}{e}\) from above as \(k\) tends to infinity.
Since \(\sum_{j=1}^{m} w_j z_j^* = Z_{LP}^* \geq \text{OPT}\), we have that
\[
E[W] \geq \left(1 - \frac{1}{e}\right)\text{OPT}
\]

- This randomized rounding algorithm can be derandomized in the standard way using the method of conditional expectations.