Randomized Methods in Computation*
Lecture 13: Randomized Rounding

Notes taken by Ido Shaked, Ariel Elbaz and David Lehmann

Spring 2001

Summary: Some approximation algorithms use linear programming on a variant of the original problem. The solutions of a linear programming problem are non-integral, in the general case, while legitimate solutions to the original problems are integers only. One idea which can be used to turn real numbers into integer solutions, is to use the linear programming (non-integer) solutions as probability parameters for selecting (integer) solutions to the original problem. In this lecture we present this idea and demonstrate it in two cases.

1 The General Idea

Given an instance of a boolean optimization problem (e.g. Max-SAT which is NP-Hard), we may reduce that problem to an analogue problem of Integer Linear Programming (ILP). Doing so, we arrive at an equivalent problem for which all the feasible solutions are integers in the range [0, 1] which means only 0,1, and such that every optimal solution for the ILP problem expresses an optimal solution for the original boolean problem. We then treat that problem as an instance of a simple Linear Programming, by dropping the integrality requirement, (note that this problem is known to be in P) and solve that problem, getting a solution expressed as Real numbers in the range [0,1]. We then use Random Rounding, a method we describe here, to convert the solution to a solution in integers 0,1 (which translates to a solution in T,F for the boolean problem), and which will have a good chance of becoming a "good" solution (one that is relatively "close" to the optimal solution).

2 Application to Max-SAT

Definition 1 Max-SAT:

Input: *A boolean formula φ over the variables $\overrightarrow{X} = (x_1, ..., x_n)$ with the following structure*

$$\varphi = \bigwedge_{j=1}^{m} C_j = \bigwedge_{j=1}^{m} (\bigvee_{i \in P_j} x_i) \vee (\bigvee_{i \in N_j} \neg x_i)$$

where N_j and P_j are the sets of negated and unnagated variables in clause C_j respectively.

Desired Output: *A truth assignment for $\overrightarrow{X}$ that satisfies the maximal number of clauses C_j.*

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

2.1 The Basic Method

We now translate an instance of Max-SAT to an analogue instance of Integer Linear Programming (ILP), as follows.

$$
\begin{array}{llll}
 & max\{\sum_{j=1}^{m} y_j\} & & \\
\text{st} \quad (i) & \forall j \in [m] & : & 0 \leq y_j \leq 1, \quad y_j \in \mathsf{Z} \\
(ii) & \forall i \in [n] & : & 0 \leq x_i \leq 1, \quad x_i \in \mathsf{Z} \\
(iii) & \forall j \in [m] & : & y_j \leq \sum_{i \in P_j} x_i + \sum_{i \in N_j} (1 - x_i).
\end{array}
$$

Any optimal solution to the ILP problem is an optimal solution to the Max-SAT problem, and vice versa.

> Oded's Note: *Looking at the ILP formulation, Condition (ii) forces the x-variables to be Boolean (or have values in $\{0,1\}$), whereas the other conditions force a correspondance between the y_j's and indicators representing whether the corresponding clauses are satisfied by the assignment to the x_i's.*

We now remove the integrality requirement and arrive at the following Linear Programming (LP) problem:

$$
\begin{array}{llll}
 & max\{\sum_{j=1}^{m} y_j\} & & \\
\text{st} \quad (i) & \forall j \in [m] & : & 0 \leq y_j \leq 1 \\
(ii) & \forall i \in [n] & : & 0 \leq x_i \leq 1 \\
(iii) & \forall j \in [m] & : & y_j \leq \sum_{i \in P_j} x_i + \sum_{i \in N_j} (1 - x_i).
\end{array}
$$

It is known that LP is solvable in polynomial time using, say, the Ellipsoid method. Thus:

Theorem 2 *LP is in P.*

Denote the optimal solution to the LP problem $\hat{X} = (\hat{x}_1, ..., \hat{x}_n), \hat{Y} = (\hat{y}_1, ..., \hat{y}_m)$.
Note that $\hat{x}_i$'s are in the range [0,1]. We will use those values as probabilities when rounding to integers 0 and 1. Suppose $\overrightarrow{Y^*} = (y_1^*, \ldots, y_n^*)$ is an optimal solution to the ILP. Clearly $\sum_{j=1}^{n} \hat{y}_j \geq \sum_{j=1}^{n} y_j^*$.

Randomized Rounding: We obtain a new integer solution by rounding: every x_i is determined by $x_i = \begin{cases} 1 & \textit{with probability } \hat{x}_i \\ 0 & \textit{otherwise} \end{cases}$.
This technique is called Randomized Rounding (R.R.).

Definition 3 *Size of Clause:* $c_j \stackrel{\text{def}}{=} |P_j| + |N_j|$.

Lemma 2.1 *Every clause C_j is satisfied, with probability at least $(1-\frac{1}{e})\cdot\hat{y}_j$, by the assignment obtained from the Randomized Rounding. That is:*

$$
\begin{aligned}
\Pr[C_j \textit{ is satisfied by } R.R.(\hat{X})] &\geq (1-(1-\frac{1}{c_j})^{c_j})\cdot\hat{y}_j \\
&\geq (1-\frac{1}{e})\cdot\hat{y}_j
\end{aligned}
$$

As a direct corollary to Lemma 2.1, we get:

Corollary 4 *The expected fraction of clauses that are satisfied by Randomized Rounding on an optimal LP solution is greater than 0.6.*

Proof: The expected number of clauses that are satisfied by Randomized Rounding is

$$
\begin{aligned}
&\sum_{j=1}^{m}\Pr[C_j \text{ is satisfied by } R.R.(\hat{X})] \\
&\geq \sum_{j=1}^{m}(1-\frac{1}{e})\cdot\hat{y}_j \\
&= (1-\frac{1}{e})\cdot\sum_{j=1}^{m}\hat{y}_j \\
&\geq (1-\frac{1}{e})\cdot\sum_{j=1}^{m}y_j^* \quad\approx\quad 0.6\cdot\sum_{j=1}^{m}y_j^*
\end{aligned}
$$

■

Proof of Lemma 2.1: Fixing any $j\in[m]$, we have.

$$
\begin{aligned}
&\Pr[C_j \text{ is not satisfied by } R.R.(\hat{X})] \\
&= \prod_{i\in P_j}\Pr[x_i=0]\cdot\prod_{i\in N_j}\Pr[x_i=1] \\
&= \prod_{i\in P_j}(1-\hat{x}_i)\cdot\prod_{i\in N_j}\hat{x}_i
\end{aligned}
$$

Define, for every i in $P_j\cup N_j$, $\hat{z}_i=\begin{cases}1-\hat{x}_i & i\in P_j\\ \hat{x}_i & i\in N_j\end{cases}$. By Condition (iii), we have

$$\hat{y}_j\leq\sum_{i\in P_j}\hat{x}_i+\sum_{i\in N_j}(1-\hat{x}_i)=\sum_{i\in P_j\cup N_j}(1-\hat{z}_i)$$

and hence

$$\sum_{i\in P_j\cup N_j}\hat{z}_i\leq c_j-\hat{y}_j.$$

Clearly

$$\begin{aligned}
&\prod_{i\in P_j}(1-\hat{x}_i)\cdot\prod_{i\in N_j}\hat{x}_i\\
&= \prod_{i\in P_j\cup N_j}\hat{z}_i\\
&\le \max_{\substack{\overrightarrow{W}=(w_1,\ldots,w_n)\\ \sum_{i\in P_j\cup N_j} w_i\le c_j-\hat{y}_j}} \left\{\prod_{i\in P_j\cup N_j} w_i\right\}
\end{aligned}$$

Note that the product of m non-negative variables, when their sum is bounded, is maximized when they are all equal. Thus, since we have $\sum_{i\in P_j\cup N_j} w_i \le c_j-\hat{y}_j$, we derive the maximum $\overrightarrow{W^*}$ to be when all w_i^* are equal; that is, $w_i^*=(\frac{c_j-\hat{y}_j}{c_j})$ for every $i\in P_j\cup N_j$. It follows that

$$\prod_{i\in P_j\cup N_j} w_i^* = (\frac{c_j-\hat{y}_j}{c_j})^{c_j} = (1-\frac{\hat{y}_j}{c_j})^{c_j}$$

So far we have proven that for every clause C_j,

$$\Pr[C_j \text{ is satisfied by } R.R.(\hat{X})] \ge 1-(1-\frac{\hat{y}_j}{c_j})^{c_j}$$

To complete our proof we need to show that

$$1-(1-\frac{\hat{y}_j}{c_j})^{c_j} \quad\ge\quad (1-(1-\frac{1}{c_j})^{c_j})\cdot\hat{y}_j \quad\ge\quad (1-\frac{1}{e})\cdot\hat{y}_j \tag{1}$$

We will use the equality : $1^c-b^c=(1-b)\cdot\sum_{i=1}^{c-1} b^i$. (*)

Specifically, applying (*) for $b=(1-\frac{\hat{y}_j}{c_j})$ and $c=c_j$, we get:

$$\begin{aligned}
1-(1-\frac{\hat{y}_j}{c_j})^{c_j} &= (1-(1-\frac{\hat{y}_j}{c_j}))\cdot[\sum_{i=1}^{c_j-1}(1-\frac{\hat{y}_j}{c_j})^i]\\
&= \frac{\hat{y}_j}{c_j}\cdot[\sum_{i=1}^{c_j-1}(1-\frac{\hat{y}_j}{c_j})^i]\\
&\ge \frac{\hat{y}_j}{c_j}\cdot[\sum_{i=1}^{c_j-1}(1-\frac{1}{c_j})^i]\\
&= \frac{1}{c_j}\cdot[\sum_{i=1}^{c_j-1}(1-\frac{1}{c_j})^i]\cdot\hat{y}_j\\
&= (1-(1-\frac{1}{c_j})^{c_j})\cdot\hat{y}_j
\end{aligned}$$

where the last equality is due to applying (*) for $b=(1-\frac{1}{c_j})$ and $c=c_j$.

This proves the first inequality in Eq. (1). The second inequality (i.e., $(1-(1-\frac{1}{c_j})^{c_j})\cdot\hat{y}_j \ge (1-\frac{1}{e})\cdot\hat{y}_j$) is derived from the fact that, for every $x\ge 1$, it holds that $(1-\frac{1}{x})^x<\frac{1}{e}$. ■

2.2 Improving the Max-SAT approximation algorithm

The above approximation algorithm achieves an approximation ratio of $(1 - \frac{1}{e})$ of the optimal solution. Yet there is a known approximation algorithm with an approximation ratio of $\frac{3}{4}$. We wish to modify the algorithm we have just seen, to achieve an approximation ratio of $\frac{3}{4}$.

Consider a random assignment for all the variables, i.e. $\overrightarrow{X} \in_R \{T, F\}^n$. Each clause C_j is not satisfied if and only if every literal in it has a false value. If we assign each variable X_i, independently, a random value from $\{T, F\}$, the probability that C_j is not satisfied is 2^{-c_j} (c_j, as before, is the number of literals in C_j), hence the probability that C_j is satisfied is $(1 - 2^{-c_j})$.

We note that for any j, when using a random assignment, the probability of C_j being satisfied increases with the size of the clause, c_j. However, using the assignment we got from solving the linear programming problem, the probability of C_j being satisfied decreases with c_j. Noting this difference calls for some combination of the two methods, to get a method that gives higher probability for each clause being satisfied regardless of it's size.

A $\frac{3}{4}$-approximation algorithm: Given a boolean formula $\varphi = \bigwedge_{j=1}^{m} C_j$ as above, we first run the previous algorithm, using Linear Programming and Randomized Rounding, and get a solution $\overrightarrow{X^1}$. We also select a random assignment $\overrightarrow{X^2} \in_R \{T, F\}^n$. We output $\overrightarrow{X^1}$ with probability $\frac{1}{2}$, $\overrightarrow{X^2}$ with probability $\frac{1}{2}$.

Analysis We claim that the expected number of satisfied clauses is at least $\frac{3}{4}$ of the optimum.

We define a new random variable v_j^1, which, for each clause C_j, will be 1 if the clause is satisfied using the assignment $\overrightarrow{X^1}$, or 0 otherwise. We use v_j^1's to write an expression for the expected number of satisfied clauses, which is simply the sum of these variables $E(\sum_{j=1}^{m} v_j^1)$, which is equal to $\sum_{j=1}^{m} E(v_j^1)$. The expected number of satisfied clauses, when using $\overrightarrow{X^1}$, is at least $\sum_{j=1}^{m}(1 - (1 - \frac{1}{c_j})^{c_j}) \cdot \hat{y}_j$, as shown above.

We also define the variables v_j^2's, for use with the assignment $\overrightarrow{X^2}$. Using $\overrightarrow{X^2}$, we've seen that for every j $E(v_j^2) = (1 - 2^{-c_j})$.

Taking each of the two assignments with probability $\frac{1}{2}$, the expected number of satisfied clauses is

$$\begin{aligned}
E(\sum_{j=1}^{m} v_j) &= \frac{1}{2} \cdot E\left(\sum_{j=1}^{m} v_j^1\right) + \frac{1}{2} \cdot E\left(\sum_{j=1}^{m} v_j^2\right) \\
&\geq \frac{1}{2} \cdot \sum_{j=1}^{m}(1 - (1 - \frac{1}{c_j})^{c_j}) \cdot \hat{y}_j + \frac{1}{2} \cdot \sum_{j=1}^{m}(1 - 2^{-c_j}) \\
&\geq \sum_{j=1}^{m} \left(\frac{1 - (1 - \frac{1}{c_j})^{c_j}}{2} + \frac{1 - 2^{-c_j}}{2}\right) \cdot \hat{y}_j \\
&\geq \sum_{j=1}^{m} \frac{3}{4} \cdot \hat{y}_j \\
&= \frac{3}{4} \cdot \sum_{j=1}^{m} \hat{y}_j
\end{aligned}$$

where the last inequality uses the following claim:

Claim 2.2 *For any positive integer x it holds that $\frac{1}{2}(1-(1-\frac{1}{x})^x)+\frac{1}{2}(1-2^{-x}) \geq \frac{3}{4}$.*

Proof: It is sufficient to prove that for any positive integer x it holds that $(1-\frac{1}{x})^x + 2^{-x} \leq \frac{1}{2}$. The claim can easily be checked for $x=1$ and 2.

> Oded's Note: *For $x=1$ we have $(1-\frac{1}{x})^x + 2^{-x} = 0+0.5 = 0.5$, whereas for $x=2$ we have $(1-\frac{1}{x})^x + 2^{-x} = 0.5^2 + 2^{-2} = 0.5$.*

For any $x \geq 3$, we have

$$(1-\frac{1}{x})^x + 2^{-x} < e^{-1} + 2^{-3} < \frac{1}{2}$$

■

3 Application to the Minimal-Set-Cover problem

Definition 5 Minimal Set Cover: *Given n and $\{S_j\}_{j=1}^m$, a collection of sets which cover $[n]$, a* **minimal set cover** *is a minimal sub-collection, J, which still covers $[n]$; that is, a minimal J for which $\bigcup_J S_j = [n]$.*

3.1 The approximation algorithm

We will go by the same path as we did for the MaxSAT approximation algorithm: first we translate the problem to an ILP problem, then solve the related LP problem, and finally use Randomized Rounding to get an Integer solution from the (non Integer) solution of the LP problem.

For every i, we name C_i to be the collection of sets that contain i. Now we rephrase the min-set-cover problem: to cover every i, we must take into J at least one of the subsets S_j that contain i, and these are all in C_i. We also should find a minimal such J. We can write the rephrased problem as an Integer Linear Programming problem:

$$\begin{array}{llll} & min\{\sum_{j\in[m]} y_j\} & & \\ \text{st} \quad (i) & \forall j \in [m] & : & 0 \leq y_j \leq 1 \quad , y_j \in \mathsf{Z} \\ \quad (ii) & \forall i \in [n] & : & \sum_{j \in C_i} y_j \geq 1 \end{array}$$

From a solution to this system corresponds to a cover $J = \{S_j \mid y_j = 1\}$. Requirement (ii) ensures that J indeed covers every i, and the minimal sum ensures the minimality of J.

We now remove the integrality requirement from the y_j's, and get a LP problem. We denote by Opt the optimal solution of the LP problem. That is, denoting the optimal LP solution by $\hat{y}_j$'s, we have $Opt \stackrel{\text{def}}{=} \sum_{j\in[m]} \hat{y}_j$.

Applying Randomized Rounding. The $\hat{y}_j$'s can be seen as a measure of the importance of S_j in the cover. However, should we, as previously in the Max-SAT problem, choose J simply by taking each S_j with probability $\hat{y}_j$, we will end up with J that has a small expected size (i.e., Opt which is at most as big as the optimum of the ILP), but will probably leave some of the i's uncovered. For example, with $C_i = \{1,2\}$ and a possible solution $\hat{y}_1 = \hat{y}_2 = 0.5$, with probabilty 1/4 none of these sets is taken (and the element i remains uncovered).

To ensure that no element remains uncover, we will take a larger random cover. Specifically, we will include S_j in J with probability $\min\{1,\ K \cdot \hat{y}_j\}$ (rather than with probability $\hat{y}_j$), where $K \stackrel{\text{def}}{=} 2\ln(2n)$.

3.2 Analysis

Claim 3.1 *The expected size of this cover is, at most, $K \cdot Opt$.*

Proof: $E(|J|) = \sum_{j=1}^{m} \Pr(j \in J) \le \sum_{j=1}^{m} K \cdot \hat{y}_j = K \cdot Opt.$ ■

Claim 3.2 *For every i,* $\Pr(i \text{ isn't covered}) \le \frac{1}{2n}$.

Proof: If, for any j in C_i, $\hat{y}_j$ is at least $\frac{1}{K}$, then S_j is always in J (because $K \cdot \hat{y}_j \ge 1$) and i is always covered. We therefore focus on the case where $\hat{y}_j < \frac{1}{K}$.

Let us define, for every j in C_i, a new random variable:

$$\eta_j = \begin{cases} 1 & \textit{with probability } K \cdot \hat{y}_j \\ 0 & \textit{otherwise} \end{cases}$$

The above is well-define because $K \cdot \hat{y}_j < 1$. Furthermore, since $K \cdot \hat{y}_j = \min(K \cdot \hat{y}_j, 1)$, the random variable η_j indicates whether or not j was taken to J. Thus, for every i in $[n]$:

$$\Pr(i \text{ isn't covered}) = \Pr\left(\sum_{j \in C_i} \eta_j = 0\right). \tag{2}$$

We will use the following Chernoff bound

A Chernoff bound: *For independent 0-1 random variables $x_1, ..., x_n$, with $\lambda = E(\sum_{i=1}^{n} x_i)$ and any $\epsilon \in [0,1]$, it holds that*

$$\Pr\left[\sum_{i=1}^{n} x_i \le (1-\epsilon)\lambda\right] \le e^{\frac{-\epsilon^2\lambda}{2}}$$

In our case, we have $\lambda \stackrel{\text{def}}{=} E(\sum_{j \in C_i} \eta_j) = K \cdot \sum_{j \in C_i} \hat{y}_j$, which by Condition (ii) is at least K. Applying the above bound (with $\epsilon = 1$), we get

$$\begin{aligned} \Pr[\sum_{j \in C_i} \eta_j = 0] &= \Pr[\sum_{j \in C_i} \eta_j \le 0] \\ &\le e^{-\lambda/2} \le e^{-K/2} \end{aligned}$$

Recalling that $K = 2\ln(2n)$, we have $e^{-K/2} = 1/2n$ and the claim follows. ■

Conclusion: We have shown that the probability that any specific i is not covered is at most $\frac{1}{2n}$. Using the union bound, it follows that the probability that there is an uncovered $i \in [n]$ is at most $\frac{1}{2}$. Thus, J is a collection of subsets S_j, with expected size no more than $2\ln(2n) \cdot Opt$, which, with probability at least $\frac{1}{2}$, covers $[n]$. Furthermore, with constant probability, J has size $O(\log n) \cdot Opt$ and covers all $[n]$.