CS602 Applied Algorithms **2019-20 Sem II**

Lecture 19: March 28

Scribe: Satyabrata Naik *Lecturer: Rohit Gurjar*

In this part of the lecture we will see the maximum satisfiability problem, and how we can use linear programming rounding to get an approximately optimal solution.

Maximum Satisfiability

Recall the satisfiability problem where we are given a boolean formula in conjunctive normal form (CNF), i.e., AND of ORs, and the goal is to find an assignment of boolean constants (0 or 1) to the variables satisfying the boolean formula, that is, satisfying all OR clauses. Maximum satisfiability is an optimization version of the problem, where given a CNF formula, we want an assignment satisfying the maximum number of clauses. For example, consider the following boolean formula.

$$f = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3)$$

The solution to above MAX-SAT is 5 (Can you do 6?). In general, this problem is NP-hard. Because we know that deciding whether there exists an assignment that satisfies all clauses is NP-hard. If we could find an assignment satisfying maximum number of clauses, that would in particular tell us whether it satisfies all clauses. So, this problem is also NP-hard and we are interested in approximation algorithms for this problem.

$\frac{1}{2}$-approx Algorithm for Maximum Satisfiability

$\frac{1}{2}$-approx algorithm gives an assignment that satisfies at least half of the number of clauses that could be satisfied by any assignment. We will consider a more general problem – maximum weight satisfiability – where each clause has a given weight and the goal is to maximize the total weight of satisfied clauses. Naturally, $\frac{1}{2}$-approx algorithm here means that it will find an assignment that achieves weight at least half of the maximum possible. Clearly, if you take all weights to be one, you get back the maximum satisfiability problem. We first look at a very simple randomized algorithm that achieves 1/2-approximation.

Randomized Algorithm:

- For each variable x_i, assign

$$\begin{aligned} x_i &= 1, \text{ with probability } \frac{1}{2} \text{ and,} \\ x_i &= 0, \text{ with probability } \frac{1}{2} \end{aligned}$$

Analysis:

Let W_C be the weight of clause C. Let W be the the total weight of the clauses that are satisfied by the above random assignment. Note that since our assignment is randomized, W is a random variable. We would like to show that the expectation of W is at least 1/2 of the maximum possible weight of satisfied clauses (OPT).

Claim 19.1. $\mathbb{E}[W] \geq \frac{1}{2}OPT$

Proof. Let us define a random variable V_C as follows:

$$V_C = \begin{cases} W_C & \text{if clause } C \text{ is satisfied by the assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

So, we have,

$$\sum_C V_C = W.$$

By linearity of expectation,

$$\begin{aligned} \mathbb{E}[W] &= \sum_C \mathbb{E}[V_C] \\ &= \sum_C W_C \cdot \mathbb{P}\text{r}[\text{Clause C is satisfied}]. \end{aligned}$$

Now, we calculate the probability term appearing in the above equation.
Consider an example of a clause,

$$C = x_{i_1} \vee \overline{x}_{i_2} \vee x_{i_3} \vee \cdots \vee x_{i_{\ell_C}}$$

Now, for this clause C,

$$\begin{aligned} \mathbb{P}\text{r}[C \text{ is not satisfied}] &= \mathbb{P}\text{r}[x_{i_1} = 0 \text{ and } x_{i_2} = 1 \text{ and } x_{i_3} = 0 \text{ and } \cdots \text{ and } x_{i_{\ell_C}} = 0] \\ &= \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \cdots \times \frac{1}{2} \ (\ell_C \textit{ times}) \\ &= \frac{1}{2^{\ell_C}} \end{aligned}$$

This tells us,

$$\begin{aligned} \mathbb{P}\text{r}[\text{C is satisfied}] &= 1 - \frac{1}{2^{\ell_C}} \\ &\geq \frac{1}{2} \quad (\text{since } \ell_C \geq 1\ ) \end{aligned}$$

Observe that this probability is better for larger clauses. Using this probability bound, we can write

$$\begin{aligned} \mathbb{E}[W] &\geq \sum_C W_C \times \frac{1}{2} \\ &\geq \frac{1}{2} OPT \qquad (\text{since} \sum_C W_C = \text{the total weight of all clauses}). \end{aligned}$$

Hence, proved. □

We can also derandomize this algorithm. We have 2^n assignments, and the expected value of total weight is at least half of maximum total weight. That means there is at least one assignment that gives at least half of maximum. We try to find such an assignment. Basic idea is to go over variables one by one, and use the self reducibility property.

Derandomization using the idea of self-reducibility:

For a weighted boolean formula f, let $W(f)$ denote the total weight of the satisfied clauses when variables are assigned randomly (0 with probability 1/2 and 1 with probability 1/2). Recall that the expectation of $W(f)$ is easy to compute by the following formula.

$$\mathbb{E}[W(f)] = \sum_{C \text{ is a clause in } f} \frac{1}{2^{\ell_C}}, \tag{1}$$

where ℓ_C is the number of literals in the clause C. The following is the deterministic algorithm to find a good assignment.

1. Assign $g \leftarrow f$ and $i \leftarrow 1$.
2. Compute g_0 and g_1 that are the formulas obtained from g by setting $x_i = 0$ and $x_i = 1$, respectively, while the weights of the clauses remain as in g.
3. Find $\mathbb{E}[W(g_0)]$ and $\mathbb{E}[W(g_1)]$ using (1).
4. If $\mathbb{E}[W(g_0)] \geq \mathbb{E}[W(g_1)]$ then assign $x_i \leftarrow 0$ and $g \leftarrow g_0$,
   otherwise assign $x_i \leftarrow 1$ and $g \leftarrow g_1$.
5. Now, if any variables are left ($i < n$), assign $i \leftarrow i + 1$ and go back to step 2.

HW: Prove that the assignment given by this deterministic algorithm achieves weight at least $\frac{1}{2}OPT$.

Integer Program and Linear Program for Maximum Satisfiability

Now we will improve the approximation factor using linear programming. Let us start by writing an integer program.

- For each variable x_i, we have an LP variable $y_i \in \{0, 1\}$, 0 denoting False and 1 denoting True.
- For each clause we have an LP variable $z_C \in \{0, 1\}$, which denotes if clause 'C' is satisfied.
- So, we want to maximize $\sum_C z_C W_C$.

To understand the constraints, take an example of a clause

$$C = x_1 \vee x_2 \vee \overline{x}_3$$

This clause is not satisfied only when x_1, x_2 are False and x_3 is true; the corresponding LP variables are $y_1 = 0$, $y_2 = 0$, $y_3 = 1$. If this is true then we would like to enforce $z_C = 0$. So, we impose the constraint

$$z_C \leq y_1 + y_2 + (1 - y_3)$$

We can easily generalize this to an arbitrary clause. Let P_C be the set of indices of positive literals and N_C be the set of indices of negative literals in the clause 'C'. Then, we have the following integer program:

$$\begin{aligned} & \max \textstyle\sum_C z_C W_C && \text{subject to} \\ z_C \leq & \sum_{i \in P_C} y_i + \sum_{i \in N_C} (1 - y_i) && \text{for each clause } C \text{ in the given formula} \\ & y_i \in \{0, 1\} && \text{for } 1 \leq i \leq n \\ & 0 \leq z_C \leq 1 && \text{for each clause } C \text{ in the given formula.} \end{aligned}$$

One can verify that this integer program captures the maximum satisfiability problem exactly. However, as we cannot solve an integer program efficiently, we need to relax this to a linear program.

$$\begin{aligned} & \max \textstyle\sum_C z_C W_C && \text{subject to} \\ z_C \leq & \sum_{i \in P_C} y_i + \sum_{i \in N_C} (1 - y_i) && \text{for each clause } C \text{ in the given formula} \\ & 0 \leq y_i \leq 1 && \text{for } 1 \leq i \leq n \\ & 0 \leq z_C \leq 1 && \text{for each clause } C \text{ in the given formula.} \end{aligned}$$

$(1 - \frac{1}{e})$-Approx Algorithm for Maximum Satisfiability

- Use a generic LP solver to find an optimal solution of the above LP, say (y^*, z^*). This solution might be fractional, so we do a rounding of the values.
- The rounding scheme will be probabilistic. Since $0 \leq y_i^* \leq 1$, we can view it as a probability. Independently, for each $1 \leq i \leq n$,

$$\begin{aligned} &\text{set } x_i = \text{True, with probability } y_i^* \text{ and} \\ &\text{set } x_i = \text{False, with probability } 1 - y_i^* \end{aligned}$$

Analysis:

We will show that the expected value of the total weight of the satisfied clauses is at least $(1 - 1/e)$ times the maximum possible weight of satisfied clauses. Continuing with the previously defined notations, we have the total weight of the satisfied clauses

$$W = \sum_{C \text{ is a clause in } f} V_C.$$

By linearity of expectation,

$$\begin{aligned} \mathbb{E}[W] &= \sum_{C \text{ is a clause in } f} \mathbb{E}[V_C] \\ &= \sum_{C \text{ is a clause in } f} W_C \cdot \Pr[\text{clause } C \text{ is satisfied}] \end{aligned} \tag{2}$$

Now, as we discussed in the first part of the lecture, we want to relate this expectation to the optimal value of the LP, which is $\sum_C z_C^* W_C$. As before, let P_C be the set of indices of positive literals, N_C be the set of indices of negative literals, and ℓ_C be the number of literals in clause C. By the rounding scheme, we can say,

$$\begin{aligned} \Pr[C \text{ is not satisfied}] &= \prod_{i \in P_C} (1 - y_i^*) \prod_{i \in N_C} y_i^* \\ &\leq \left(\frac{\sum_{i \in P_C} (1 - y_i^*) + \sum_{i \in N_C} y_i^*}{\ell_C} \right)^{\ell_C} \end{aligned} \tag{3}$$

The inequality comes from the $AM \geq GM$ inequality, which states that given positive numbers α_i for $1 \leq i \leq k$,

$$\left(\sum_{i=1}^{k} \alpha_i \right) / k \geq \left(\prod_{i=1}^{k} \alpha_i \right)^{1/k}.$$

From the constraints of the LP, we have

$$\begin{aligned} & z_C^* \leq \sum_{i \in P_C} y_i^* + \sum_{i \in N_C} (1 - y_i^*) \\ \implies \quad & -z_C^* \geq \sum_{i \in P_C} (-y_i^*) + \sum_{i \in N_C} (y_i^* - 1) \\ \implies \quad & -z_C^* \geq \sum_{i \in P_C} (-y_i^* + 1 - 1) + \sum_{i \in N_C} (y_i^* - 1) \\ \implies \quad & -z_C^* \geq \sum_{i \in P_C} (1 - y_i^*) + \sum_{i \in N_C} (y_i^*) - \ell_C \\ \implies \quad & \ell_C - z_C^* \geq \sum_{i \in P_C} (1 - y_i^*) + \sum_{i \in N_C} (y_i^*) \end{aligned}$$

Using this with (3), we obtain

$$\begin{aligned} \mathbb{P}\mathrm{r}[C \text{ is not satisfied}] &\leq \left(\frac{\ell_C - z^*_C}{\ell_C}\right)^{\ell_C} = \left(1 - \frac{z^*_C}{\ell_C}\right)^{\ell_C} \\ \implies \mathbb{P}\mathrm{r}[C \text{ is satisfied}] &\geq 1 - \left(1 - \frac{z^*_C}{\ell_C}\right)^{\ell_C} \end{aligned}$$

Using (2), we can write

$$\mathbb{E}[W] \geq \sum_C W_C \left(1 - \left(1 - \frac{z^*_C}{\ell_C}\right)^{\ell_C}\right). \tag{4}$$

Recall that we want to compare the above quantity with the LP optimal $\sum_C W_C z^*_C$. Which just means that we need to compare the quantity $1 - (1 - z^*_C/\ell_C)^{\ell_C}$ with z^*_C. We show the following.

Claim 19.2. *For any $0 \leq z \leq 1$ and $\ell \geq 1$,*

$$1 - (1 - z/\ell)^\ell \geq (1 - 1/e)z.$$

Proof. Consider the function $h(z) = 1 - (1 - z/\ell)^\ell$. If $\ell = 1$, then $h(z) = z$ and thus the claim follows immediately. For $\ell \geq 2$, we argue that $h(z)$ is a concave function. To see this, simply consider the second derivative of $h(z)$.

$$\begin{aligned} h''(z) &= -\frac{1}{\ell^2}(1 - z/\ell)^{\ell-2}. \\ &\leq 0, \quad (\text{ since } z/\ell \leq 1). \end{aligned}$$

Now, by concavity of $h(z)$, we know for any $0 \leq z \leq 1$,

$$\begin{aligned} h(z) &\geq (1 - z)h(0) + zh(1) \\ &= 0 + z(1 - (1 - 1/\ell)^\ell) \\ &\geq z(1 - 1/e). \end{aligned}$$

□

Using the above claim with (4), we get,

$$\mathbb{E}[W] \geq (1 - 1/e) \sum_C W_C z^*_C = (1 - 1/e)\ \mathrm{OPT(LP)}.$$

Finally, we know that since the linear program is a relaxation of the integer program, OPT(LP) must be at least the actual OPT – the maximum possible weight of the satisfied clauses for any assignment. Thus,

$$\mathbb{E}[W] \geq (1 - 1/e)\ \mathrm{OPT}.$$

This shows that the algorithm achieves $(1 - 1/e)$-approximation.

Derandomization using the idea of self-reducibility:

We can use exactly the same idea for derandomization, as was used in the simple 1/2-approximation algorithm. The difference is the expression we will use for the expected weight of the satisfied clauses.

$$\mathbb{E}[W(f)] = \sum_{C \text{ is a clause in } f} W_C \cdot \left(1 - \prod_{i \in P_C}(1 - y^*_i) \prod_{i \in N_C} y^*_i\right)$$

Given y^*, this expected weight is easy to compute for f or any of its sub-formulas (after setting some variables).

HW: Prove that there is a deterministic version of the $(1 - 1/e)$-factor algorithm.

Better Approximation Factor than $1 - 1/e$: A Combination of the Two Schemes

Can we improve the approximation factor from $(1 - 1/e) \approx 0.63$? Let us compare the analysis of the two randomized algorithms we have seen. In the first algorithm, for a clause C of length ℓ_C, we had

$$\mathbb{P}\text{r}[C \text{ is satisfied}] = 1 - 1/2^{\ell_C}.$$

This probability increases as the clause length ℓ_C increases. For the analysis we had considered the worst case which was $\ell_C = 1$. On the other hand, in the second algorithm, it can be seen from the analysis that

$$\mathbb{P}\text{r}[C \text{ is satisfied}] \geq z^*_C(1 - (1 - 1/\ell_C)^{\ell_C}).$$

Here, the factor $(1 - (1 - 1/\ell_C)^{\ell_C})$ actually decreases as ℓ_C increases. And the worst case considered was $\ell_C = \infty$. In short, the first scheme was good for larger clauses and the second scheme seems to be good for smaller clauses. So, we can try to work with a convex combination of the two ideas.

Recall that a boolean variable x_i was set to be True with probability $1/2$ in the first scheme and with probability y^*_i in the second scheme. Now, we set x_i to be True with probability

$$1/4 + y^*_i/2.$$

You can do a similar analysis and get

$$\mathbb{P}\text{r}[C \text{ is satisfied}] \geq z^*_C \left(1 - \left(\frac{3}{4} - \frac{1}{2\ell_C}\right)^{\ell_C}\right).$$

You can show that the multiplicative factor here is at least $3/4$ for all values of ℓ_C and get

$$\mathbb{P}\text{r}[C \text{ is satisfied}] \geq (3/4)z^*_C$$

This will imply that you have a $(3/4)$-approximation algorithm. We leave the details as exercise. One can also derandomize this algorithm using the same ideas.

You might wonder if there is a different convex combination of the two schemes that can give a better bound. You can consider the following general scheme with some $0 \leq \alpha \leq 1$: set x_i to be True with probability

$$(1 - \alpha)/2 + \alpha y^*_i.$$

You can do the analysis and verify that, in fact, taking $\alpha = 1/2$ gives you the best bound, as we did above.

Further Improvement on the Approximation Factor?

We saw that a clever combination of two schemes gave us a better rounding scheme. Is it possible to design a better rounding scheme? The answer turns out to be no. One can associate the so called *integrality gap* for any LP relaxation (of any problem), that puts a limit on how good approximation factor you can get, no matter what rounding scheme you are using. Figure 1 shows relative positions of three values on the real line: the LP optimal (LP-OPT), the actual optimal (OPT) which is also the integral optimal, and the value of the rounded solution (ROUNDED) that we output. The approximation factor is given by the ratio ROUNDED/OPT. However, since we have no clue about the actual optimal OPT, we instead bound the ratio ROUNDED/LP-OPT that lower bounds the desired ratio ROUNDED/OPT. But, how good a bound can we obtain on ROUNDED/LP-OPT? Certainly it cannot be larger than OPT/LP-OPT. The worst possible gap between OPT and LP-OPT is called the integrality gap (shown by the green bar in Figure 1). Formally,

$$\text{Integrality gap } = \min_{f \text{ is a CNF formula}} \frac{\text{OPT}(f)}{\text{LP-OPT}(f)}$$

And as discussed above, the analysis we use cannot guarantee a better approximation factor than the integrality gap.

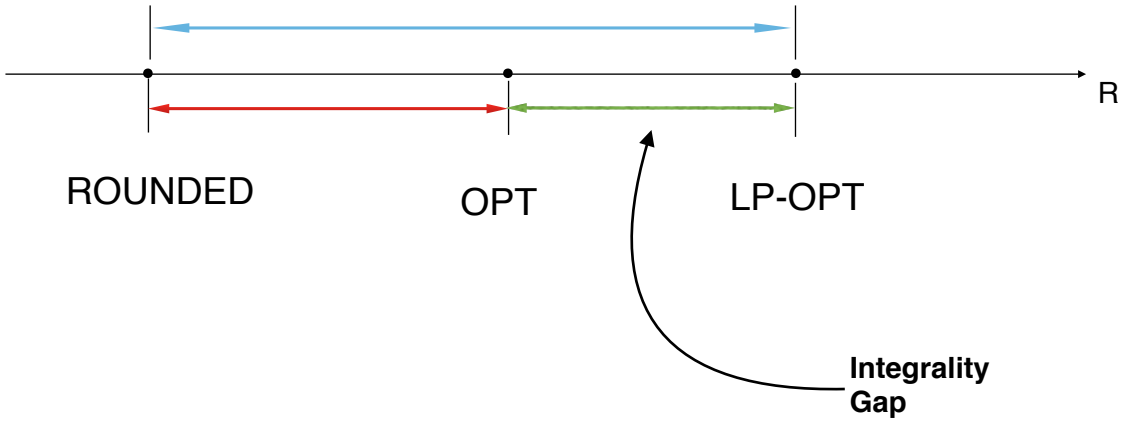

Figure 1: LP optimal, LP integral optimal and the value of the rounded solution. The red bar shows the gap we want to upper bound and the blue bar shows the gap we are actually going to upper bound. The green bar shows the integrality gap.

Integrality gap for max-satisifiability LP

We now show an integrality gap of 3/4 for the LP relaxation we used for maximum satisfiability. Note that integrality gap is defined to be the worst possible ratio over all possible inputs. We will just show one input example which has the gap of 3/4 between OPT and LP-OPT. This will mean that no matter what rounding scheme we use, our analysis cannot give an approximation factor better than 3/4. Since we also have a (3/4)-approximation algorithm, it means that this example is the worst possible.

Example with 3/4 **integrality gap.** Consider the Boolean formula:

$$f = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_2).$$

Assume weight 1 for each clause. Clearly, you can satisfy at most 3 out of these 4 clauses. Thus, OPT=3. On the other hand, the LP-OPT value is 4 (left as an exercise).

Interestingly, this example is not bad for the algorithms we have seen, because any assignment will satisfy 3 clauses which is the OPT. So, it is possible for some particular problem instance, that the actual approximation factor is better than the integrality gap. It is just that we cannot get a better guarantee via the analysis we do. To show that 3/4 is really the best possible factor for max satisfiability achievable via LP rounding, do the following.

HW: Find an example of a CNF formula, where ROUNDED $\approx$ 3/4 (OPT).