LP Rounding and Randomized Quicksort

1 LP Rounding Algorithm for Max-SAT

1.1 LP Rounding

Assume we are given an arbitrary CNF formula with different clause lengths and weights w_j on the clauses. Then we'd like to reformulate the maximization problem as an integer linear program, relax it to a fractional version, and round the fractional version.

Let y_j be an LP variable for each clause C_j, with value 1 if C_j is true and 0 otherwise. Also, let z_i be an LP variable for each variable x_i, with value 1 if x_i is true and 0 if not. Then we have the LP:

$$\begin{aligned}
\max \quad & \sum_j w_j y_j \\
\text{s.t.} \quad & y_j \leq \sum_{x_i \in C_j} z_i + \sum_{\bar{x}_i \in C_j} 1 - z_i & \forall j \\
& 0 \leq y_j \leq 1 & \forall j \\
& 0 \leq z_i \leq 1 & \forall i
\end{aligned}$$

This LP can be solved in polynomial time. Note that $OPT_{ILP} \leq OPT_{LP}$ because each legal solution to the integer linear program is also a legal solution to the fractional LP.

For our rounding scheme, we will independently set each variable x_i to true with probability z_i. Let

$$Y_j = \begin{cases} 1 & \text{if } C_j \text{ is true} \\ 0 & \text{otherwise} \end{cases}.$$

Thus $Y = \sum_j w_j Y_j$. So,

$$\begin{aligned}
\mathbb{E}[Y] &= \sum_j w_j \mathbb{E}[Y_j] \\
&= \sum_j w_j \Pr[C_j \text{ true}].
\end{aligned}$$

By independence,

$$\begin{aligned}
\Pr[C_j \text{ true}] &= 1 - \Pr[C_j \text{ false}] \\
&= 1 - \prod_{x_i \in C_j} (1 - z_i) \cdot \prod_{\bar{x}_i \in C_j} z_i.
\end{aligned}$$

For analysis of just one clause, we can assume for simplicity (by renaming x_i to $\bar{x}_i$ for some i) that all literals are un-negated. So,

$$\Pr[C_j \text{ true}] = 1 - \prod_{x_i \in C_j} (1 - z_i).$$

The LP solution satisfies $y_j \le \sum_{x_i \in C_j} z_i$ for all j. If $\sum_{x_i \in C_j} z_i$, that increases our probability of satisfying C_j without helping the LP. So the worst case is $\sum_{x_i \in C_j} z_i = y_j$. Subject to this equality, the best way to set the z_i is as unequal as possible, and the worst would be to set them all equal (by convexity).

Write $k = |C_j|$. Then in the worst case, $z_i = Y_j/k$. So,

$$\Pr[C_j \text{ true}] \ge 1 - \big(1 - \frac{Y_j}{k}\big)^k$$

We want to lower bound this by αY_j for some large constant α. To do so, we will prove the function is concave over $[0, 1]$ and then lower bound at the endpoints $y_j = 0$ and $y_j = 1$. This will result in a lower bound for the entire range.

The first derivative is

$$k\big(1 - \frac{Y_j}{k}\big)^{k-1} \ge 0,$$

and the second derivative is

$$\frac{-(k-1)}{k}\big(1 - \frac{Y_j}{k}\big)^{k-2} \le 0,$$

proving monotonicity and concavity. At $y_j = 0$, the function is equal to 0, and at $y_j = 1$, the function is equal to $1 - (1 - 1/k)^k \ge 1 - 1/e$. Then,

$$\Pr[C_j \text{ true}] \ge \big(1 - \frac{1}{e}\big)y_j.$$

Thus we have shown that the expected weighted number of clauses satisfied is at least

$$\begin{aligned}\sum_j w_j \Pr[C_j \text{ satisfied}] &\ge \sum_j w_j(1 - 1/e)y_j \\ &= (1 - 1/e)OPT_{LP} \\ &\ge (1 - 1/e)OPT_{ILP}.\end{aligned}$$

So this is a $(1-1/e)$ approximation. Johnson's algorithm does better for $k \ge 2$, so our LP rounding algorithm only helps if we are guaranteed to have singleton clauses.

1.2 A Hybrid Algorithm

A better algorithm then either alone is to run both algorithms and keep the better output; for our analysis we will flip a fair coin to decide which algorithm to run. Note that this strategy can only do worse than keeping the better. Then, the expected weight of satisfied clauses is

$$\begin{aligned}\frac{1}{2}\big(\mathbb{E}[\text{Johnson}] + \mathbb{E}[\text{LP}]\big) &= \frac{1}{2}\sum_j w_j\big(\Pr[\text{Johnson satisfies } C_j] + \Pr[\text{LP-rounding satisfies } C_j]\big) \\ &\ge \frac{1}{2}\sum_j w_j\big((1 - 2^{-|C_j|}) + (1 - (1 - \frac{1}{|C_j|})^{|C_j|}y_j)\big).\end{aligned}$$

If $|C_j| = 1$, then this is $\ge 1/2 + y_j$. If $|C_j| = 2$ then this is $\ge 3/4 + 3/4y_j$. Finally for $|C_j| \ge 3$ this is $\ge 7/8 + (1 - 1/e)y_j$. Note that a lower bound for all these cases is $3/2 \cdot y_j$. Thus we obtain

$$\begin{aligned}\frac{1}{2}\big(\mathbb{E}[\text{Johnson}] + \mathbb{E}[\text{LP}]\big) &\ge \frac{1}{2}\sum_j \frac{3}{2}w_j \cdot y_j \\ &= \frac{3}{4}OPT_{LP}.\end{aligned}$$

So this is a 3/4 approximation.

2 Randomized Quicksort

Recall that Quicksort(S) works as follows:

1. Pick a pivot $x \in S$.
2. Divide S into $S_1 = \{y \in S : y \leq x\}$ and $S_2 = \{y \in S : y > x\}$.
3. Quicksort(S_1); Quicksort(S_2).
4. Output sorted S_1 followed by sorted S_2.

The runtime recurrence is

$$T(|S|) \leq T(|S_1|) + T(|S_2|) + O(|S|),$$

where $O(|S|)$ is the runtime of the partition step. This has solution

$$T(|S|) \leq O(|S| \cdot \log |S|)$$

if the partition is always (most of the time) such that $|S_1|, |S_2| \leq \alpha \cdot |S|$ for some constant $\alpha < 1$. In particular, this works well if x is the median, and then we obtain $\alpha = 1/2$. If the pivot is always chosen poorly (e.g., $|S_2| = 1$), then runtime is $\Omega(|S|^2)$.

For Randomized Quicksort, we just pick $x \in S$ uniformly at random. Conceptually, this works because a "good pivot" is somewhere between the first and third quantiles of the sorted array, and if we pick such a pivot then we get $\alpha \leq 3/4$. We have a 1/2 chance to pick a "good pivot", so we make enough progress 1/2 the time, and that is enough. This can be turned into a formal proof, but we'll focus on comparisons only.

Let Y be the number of total comparisons. We want to bound $\mathbb{E}[Y]$. Let

$$Y_{ij} = \begin{cases} 1 & \text{if } i,j \text{ are ever compared} \\ 0 & \text{otherwise} \end{cases}.$$

Then, $Y = \sum_{i<j} Y_{ij}$. So,

$$\begin{aligned} \mathbb{E}[Y] &= \sum_{i<j} \mathbb{E}[Y_{ij}] \\ &= \sum_{i<j} \Pr[i,j \text{ compared}]. \end{aligned}$$

So we just need to calculate $\Pr[i,j \text{ compared}]$. Let's number the elements in sorted order. Then i,j can only be compared if/when one if a pivot at a time when both $i,j \in S$. Note that if an element x between i,j is ever chosen as a pivot before i or j, then i and j are separated forever and will never be compared. Thus, we want to calculate the probability that such an x is chosen before i or j.

The moment the first element from $\{i,\ldots,j\}$ is chosen as a pivot is when Y_{ij} is decided. If i or j is chosen, then $Y_{ij} = 1$. If $x \notin \{i,j\}$, then $Y_{ij} = 0$.

We have that $\Pr[Y_{ij} = 1] = 2/(j+1-i)$ because there are $j+1-i$ elements in the range, or which 2 cause a comparison. Conditioned on picking the first element from $\{i,\ldots,j\}$, the choice is uniformly random. So,

$$\mathbb{E}[Y] = \sum_i \sum_{j>i} \frac{2}{j+1-i}$$

Let $k = j + 1 - i$. Then $j = k + i - 1$, and

$$\begin{aligned}
\mathbb{E}[Y] &= \sum_{i} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&\leq 2 \sum_{i} \sum_{k=1}^{n} \frac{1}{k} \\
&= 2nH(n) \\
&= \Theta(n \log n)
\end{aligned}$$

This is a Monte Carlo algorithm with expected runtime $\mathcal{O}(n \log n)$. One can also prove high-probability bounds.