CS 388R: Randomized Algorithms, Fall 2021 Nov 23, 2021

Lecture 25: Randomized Rounding

Prof. Eric Price *Scribe: Connor Colombe*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In this lecture we look at how we can use the concept of randomized rounding to develop approximation algorithms. To get a better sense of what exactly this means, let's jump into an example using the weighted set cover problem.

2 Weighted Set Cover

Suppose we are given n items that form a universe of items U, m sets of items $S_1, \ldots, S_m \subset U$, and weights $w_1, \ldots, w_m$ for each set.

Definition: We call a set of indices $C \subseteq [m]$ a *cover* if

$$\cup_{i \in C} S_i = U.$$

In other words, for each of our item j, there is some i in our cover C such that S_i contains j. The goal of the weighted set cover problem is to find a cover C for our n items using a collection of S_i with minimal total weight/cost. Let the cost of a cover C be given by

$$Cost(C) = \sum_{i \in C} w_i$$

Unfortunately, this problem is NP-Hard. So how ought we go about solving it? Let's use an approximation algorithm! Instead, of solving the problem exactly, we will now search for a cover C such that

$$Cost(C) \leq \alpha \cdot OPT$$

where OPT is the minimal possible cost of a cover for our sets and $\alpha \geq 1$ is the approximation factor.

To begin, we will first cast the weighted set cover problem as an integer program (IP). An integer program is an optimization problem where the decision variables are restricted to be integral. In our case, the IP for weighted set cover is given by

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} w_i x_i \\ & \sum_{i:S_i \ni j} x_i \geq 1 \quad \forall j = 1, 2, \ldots, n \\ & x_i \in \{0, 1\} \quad \forall i = 1, 2, \ldots, m \end{aligned}$$

where x_i is the binary decision variable that takes on unit value if index i is in the cover C. Observe that the first constraint enforces that any feasible $x = (x_1, \ldots, x_m)$ is a cover. Let the minimum possible cover weight be denoted OPT.

Since solving this IP is NP-Hard, we will instead relax the integrality constraints by allowing each x_i to take on any value on the unit interval $[0, 1]$. Relaxing this constraint, we can now formulate the linear programming (LP) relaxation as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} w_i x_i \\ & \sum_{i:S_i \ni j} x_i \geq 1 \quad \forall j = 1, 2, \ldots, m \\ & 0 \leq x_i \leq 1 \quad \forall i = 1, 2, \ldots, m \end{aligned}$$

Fortunately, LP's are solvable in polynomial time! However, upon solving the LP relaxation, we may find that our LP solution is not feasible in the original IP as some decision variables may have fractional values.

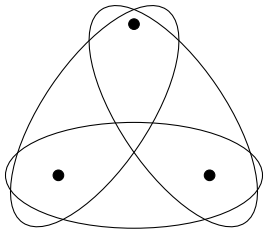

Figure 1: An example of an instance where the optimal IP solution is $OPT = 2$ achieved by picking any two sets. However, the LP relaxation assigns $x_i = 1/2$ which gives a total weight of $3/2$.

If we consider the example of the small figure above, the optimal solution gives $OPT = 2$. However, we can see that by assigning $x_i = 1/2$ for each set, we can find a cover of total weight $3/2$ which is optimal for the LP relaxation. Denote the optimal solution to the LP, OPT_{LP}. Note that since we are relaxing a constraint (and thus making the solution space larger), it must be the case that

$$OPT_{LP} \leq OPT$$

for any LP relaxation of a minimizing IP.

So we would like to go from this fractional solution to a feasible integer solution to our IP. How ought we go about doing this? One way is to use randomized rounding. The idea is to first solve the LP relaxation to get a solution $x = (x_1, \ldots, x_m)$. Then for each x_i randomly round x_i to 0 or 1 with probability x_i to form x_i^*, and let x^* be our new candidate solution to the weighted set cover problem.

We see that in expectation

$$\mathbb{E}[Cost(C)] = \mathbb{E}\left[\sum_{i \in C} w_i\right] = \sum_{i \in [m]} x_i w_i = OPT_{LP}.$$

Unfortunately this random rounding won't always produce a valid cover to our original problem.

Question: What is the probability that j is not covered by our rounded C?

$$\begin{aligned}
\mathbb{P}(\text{No set that contains } j \text{ was put in } C) &= \prod_{i:S_i \ni j} (1 - x_i) \\
&\leq \prod_{i:S_i \ni j} e^{-x_i} \\
&= e^{-\sum_{i:S_i \ni j} x_i} \\
&\leq 1/e
\end{aligned}$$

where in the first line we use the inequality $(1 - x) \leq e^{-x}$ and in the third that $\sum_{i:S_i \ni j} x_i \geq 1$ since this is a constraint in our LP relaxation. This implies that each item has at least a $1 - 1/e$ chance of being covered with this randomized rounding. We can reason that if we randomly round our x in this manner $T = \ln 4n$ times to produce the the potential covers $C_1, \ldots, C_T$, we can then output $C^* = \cup_{t=1}^{T} C_t$. We get that

$$\mathbb{E}[Cost(C^*)] \leq T \cdot OPT_{LP}$$

since each C_i has expected value OPT_{LP} and in the worst case all T of them are disjoint. Note

$$\mathbb{P}[C^* \text{ not a cover}] \leq ne^{-T} \leq 1/4.$$

So half of the time $COST(C^*) \leq 4T \cdot OPT_{LP}$ and C^* is a cover.

3 Max-SAT

Another example of how randomized rounding can be useful is in developing an approximation algorithm for Max-SAT. The premise of the problem is that given a set of clauses

$$(x_1 \vee \overline{x_2}) \wedge (x_2 \vee x_3 \vee \overline{x_4}) \cdots$$

how many of them can we satisfy at once? More formally, let there be m clauses of the form $C_j = (C_j^+, C_j^-)$ with length ℓ_j and there be n variables $x_1, x_2, \ldots x_n$. What is the maximum number of clauses that can be satisfied? We will begin by considering an instance of 3-SAT ($\ell_j = 3$ for all j).

With the problem stated, how do we go about developing an approximation algorithm? A naive approach would be to randomly pick $x_i \in \{0, 1\}$ iid uniform. Under this assignment, the probability that any clause j is satisfied is

$$\mathbb{P}(\text{clause } j \text{ is satisfied}) = 1 - \frac{1}{2^{\ell_j}} = 1 - \frac{1}{2^3} = 7/8$$

which then implies that $\mathbb{E}[\#\text{ of clauses satisfied}] = \frac{7}{8}m \geq \frac{7}{8} \cdot OPT$.

Question: Can we get a $\geq \frac{7}{8}m$ algorithm in polynomial time? Indeed we can. The idea is for each variable x_i compute the expected number of clauses satisfied given $x_i = 0$ and then given $x_i = 1$. Next, set x_i to the value with the larger expectation (one will always be larger than $\frac{7}{8}m$) and repeat until we have found a solution. Note this is actually no longer a randomized algorithm! It is an instance of a deterministic algorithm inspired by a randomized one. In the more general case of Max-Sat (ℓ_j no longer restricted to 3), this randomized assignment strategy achieves

$$\mathbb{E}[\#\text{ of clauses satisfied}] = \sum_j 1 - \frac{1}{2^{\ell_j}} \geq m/2.$$

Now we consider an algorithm that uses the IP/LP relaxation as before. Let y_j take on unit value if the j-th clause C_j is satisfied. The IP formulation of Max-SAT is given by

$$\begin{aligned}
\max \quad & \sum_{j=1}^{m} y_j \\
& y_j \leq \sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} 1 - x_i \quad \forall j = 1, 2, \ldots, m \\
& x_i \in \{0, 1\} \quad \forall i = 1, 2, \ldots, n \\
& y_j \in \{0, 1\} \quad \forall j = 1, 2, \ldots, m
\end{aligned}$$

For the LP relaxation, we now allow x_i and y_j to be any value on the unit interval. Solve the LP relaxation to get solution x. As before we are going to use randomized rounding. Let $\mathbb{P}(x_i^* = 1) =$

x_i. The probability that clause j is *not* satisfied is given by

$$\begin{aligned}
\mathbb{P}(\text{No set that contains } j \text{ was put in } C) &= \mathbb{P}(\text{no } x_i^* = 1 \; \forall i \in C_j) \\
&= \prod_{i \in C_j} (1 - x_i) \\
&\leq e^{-\sum x_i} \\
&\leq e^{-y_j} \\
&\leq 1/e
\end{aligned} \tag{1}$$

This implies that the expected number of clauses satisfied is $(1 - 1/e) \cdot OPT \approx 0.63 \cdot OPT$. This is better than the $m/2$ from the random assignment!

However this is not as good as the randomized assignment for Max-3-SAT which is a little unsatisfying. The issue lies in the approximation used in the third line of (1). It isn't tight enough. Note that by the arithmetic mean-geometric mean inequality, we have that

$$\prod_{i \in C_j} (1 - x_i) \leq \left(1 - \frac{\sum_{i \in C_j} x_i}{\ell_j}\right)^{\ell_j} \leq \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}.$$

Observe that in the limit $\ell_j \to \infty$, we recover the $1/e$ bound.

If we plot the expected approximation ratios for each algorithm as a function of clause size, we get Table 1.

ℓ_j	probabilistic rounding:$1 - (1 - 1/\ell_j)^{\ell_j}$	uniform rounding: $1 - 2^{-\ell_j}$
1	1	.5
2	.75	.75
3	.7	.875
4	.683	.9375
$\vdots$	$\vdots$	$\vdots$
∞	$1 - 1/e$	1

Table 1: The expected approximation factors for the proportional random rounding and uniform random rounding as a function of clause size.

Note that there is a trade-off in clause size and the approximation ratio! If we are presented with a problem that has both long and short clauses what should we do? In practice we ought to compute both $\sum_j 1 - (1 - 1/\ell_j)^{\ell_j}$ and $\sum_j (1 - 2^{-\ell_j})$ and use the algorithm with the better approximation. However, if we randomly pick which method to use, we have that

$$\mathbb{P}(C_j \text{ is satisfied}) = \frac{1}{2}\left(1 - (1 - 1/\ell_j)^{\ell_j}\right) + \frac{1}{2}\left(1 - 2^{-\ell_j}\right) \geq \frac{3}{4}$$

which implies that regardless of clause sizes, we can always achieve at least an expected $3/4 \cdot OPT$ approximation ratio.