**Approximation Algorithms** **Spring 2019**

# Lecture 1: Intro to Approximation Algorithms: Set Cover

Scribe: Jiazheng Zhao 2/8/2019

## 1.1 Introduction

Suppose $\mathbf{P} \neq \mathbf{NP}$, then it is impossible to find optimal solutions for many discrete optimization problems such as Set Cover, Traveling Salesman, and Maxcut efficiently. The study of approximation algorithms focuses on developing algorithms that relax the requirement of finding an optimal solution and instead searches for those that are "good enough" *efficiently* given *any* instance. Furthermore, it studies providing mathematically rigorous methods to quantify what "good enough" means. We will often measure an approximation algorithm's efficacy by its *approximation ratio*.

**Definition 1.1.** *An $\alpha$-approximation algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a result whose value is within a factor of $\alpha$ of the value of the optimal solution. The algorithm's approximation ratio is then given by $\alpha$.*

If we are dealing with a maximization problem, we expect $\alpha < 1$. Similarly, we would expect $\alpha > 1$ for a minimization problem. In these notes, we will provide an overview of iconic techniques used in the design of approximation algorithms by looking at an example problem: Set Cover.

## 1.2 Set Cover

In the set cover problem, we are given a set of elements $E = \{e_1, e_2, \ldots, e_n\}$ and a collection of subsets $S_1, S_2, \ldots, S_m \subseteq E$ where each is associated with weights $w_1, \ldots, w_m \geq 0$ of each subset $S$. The goal of this problem is to find a collection of subsets $\{S_j\}_{j \in I}$ such that $\cup_{j \in I} S_j = E$ while simultaneously minimizing the sum of weights $\sum_{j \in I} w_j$.

Set cover is one of Karp's 21 NP-complete problems and unless $\mathbf{P} \neq \mathbf{NP}$ it is unlikely we can find an optimal solution efficiently for all instances. Instead, we will focus on designing various algorithms to approximate Set Cover. We will first discuss a greedy algorithm for Set Cover, then demonstrate two algorithms that rely on the linear program reduction of Set Cover. The first rounds the solution of the LP deterministically, while the second randomly. Finally, we show an algorithm that rounds the dual LP for Set Cover.

## 1.3 A Greedy Algorithm

A greedy algorithm iteratively builds a solution by making a sequence of myopic decisions; each decision optimizes an often simple heuristic though doing so may construct a solution that deviates from the optimum. Let's consider a simpler version of Set Cover where all sets $S_j$ have weight $w_j = 1$. If we are to construct a cover iteratively, then a reasonable heuristic to optimize for is to always pick the set that includes the greatest number of currently uncovered elements. Our greedy approximation algorithm is the following:

**Greedy Algorithm**
Given $E = \{e_1, \ldots, e_n\}$ and $S_1, \ldots, S_m \subseteq E$ initialize $I = \varnothing$. While $I$ does not cover $E$:

1. For each $j$, let $\hat{S}_j$ be the elements of $S_j$ currently uncovered by sets in $I$.
2. Compute $\ell = \text{argmax}_{j:\hat{S}_j \neq \varnothing} |\hat{S}_j|$.
3. Update $I \leftarrow I \cup \{\ell\}$

Let us now analyze the efficacy of this algorithm.

**Theorem 1.2.** *The greedy algorithm produces a* $\ln n$*-approximation algorithm for the Set Cover problem.*

What does it mean to be a $\ln n$-approximation algorithm for Set Cover? The goal of Set Cover seeks to minimize the sum of set weights, or just the number of sets chosen because we assume $w_j = 1$. The claim states that the algorithm returns a solution that is at most $\ln n$ factor greater than the optimal sum of weights. Suppose that $I^*$ denotes the optimal set cover and $I$ denotes the set cover returned by the algorithm. Then a $\ln n$-approximation guarantees that

$$\sum_{j \in I^*} w_j \leq \sum_{j \in I} w_j \leq \ln n \cdot \sum_{j \in I^*} w_j$$

This often written more succinctly as

$$\text{OPT} \leq \sum_{j \in I} w_j \leq \ln n \cdot \text{OPT}$$

We can now use this to demonstrate theorem 1.2.

*Proof.* Let $n_t$ denote the number of elements not covered after $t$ iterations of the greedy algorithm. At iteration $t = 0$, all elements remain to be covered hence $n_0 = n$. Suppose that $I^*$ is the optimal set cover with $k$ elements. At any iteration $t$, there always exists a set $S_j$ such that it covers at least $\frac{n_t}{k}$ elements. This is because $I^*$, being a set cover, must cover all remaining $n_t$ elements. If no set covers $\frac{n_t}{k}$ elements then no combination of $k$ sets can cover the remaining $n_t$ elements. It could not be that $I^*$ is a set cover.

Since there is a set covering $\frac{n_t}{k}$ elements at every time step $t$, then the greedy algorithm will at least be able to cover $\frac{n_t}{k}$ additional elements. Hence

$$n_{t+1} \leq n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right) \leq n_0\left(1 - \frac{1}{k}\right)^t$$

However, $1 - x \leq e^{-x}$ and so

$$n_t \leq n_0\left(1 - \frac{1}{k}\right)^t < n_0(e^{-1/k})^t = n_0 e^{-t/k}$$

If $t = k \ln n$, then $n_t < 1$ meaning there are no more elements to cover. Each iteration adds one set and so the size of the returned set cover $I$ will be $|I| \leq k \ln n$. However, $k = |I^*|$ and so $|I| \leq |I^*| \ln n = \ln n \cdot \text{OPT}$ as required. □

This algorithm turns out to be almost as good as we can achieve assuming $\mathbf{P} \neq \mathbf{NP}$.

**Theorem 1.3.** *There exists some constant $c > 0$ such that if there exists a $c \ln n$-approximation algorithm for the unweighted set cover problem, then $\mathbf{P} = \mathbf{NP}$.*

## 1.4 Linear Programming Solutions

Linear programming is another versatile tool for the design and analysis of approximation algorithms. Recall that a linear program is a optimization problem of the form

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^\top \boldsymbol{x} \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \end{aligned}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. A similar system can be written for maximization problems and we can append a constraint $x_i \in \{0, 1\}$ to make this into an *integer* linear program. Many discrete optimization problems can be formulated into integer linear programs where the vector solution encodes the combinatorial solution. However, integer linear programs are in general NP-Hard to solve and so LP based approximation algorithms take the following form to get around this:

1. Write the integer linear program for the discrete optimization problem.
2. Relax the constraint $x_i \in \{0, 1\}$ to $x_i \geq 0$.
3. Solve the linear program using a method such as simplex, interior point, etc.
4. Take the continuous solution $\boldsymbol{x}$ and apply a rounding procedure that converts components into integers $\hat{\boldsymbol{x}}$ and return $\hat{\boldsymbol{x}}$.

We will now see how to apply this procedure to Set Cover.

### 1.4.1 Formulating the LP

First, we have to write an integer linear programming for Set Cover. Begin by considering what we want our solution vector $x$ to be. We want $x$ to encode whether or not to include a set $S_j$ in the final set cover so let

$x_j = 1$ if $S_j$ should be included in the cover and $x_j = 0$ otherwise.

$$x_j = \begin{cases} 1 & S_j \text{ is in solution} \\ 0 & \text{otherwise} \end{cases}$$

We want to minimize the sum of weight of subsets we pick, therefore our objective is:

$$\text{minimize} \sum_{j=1}^{m} w_j x_j$$

Finally, we want all elements to be a part of the set cover. We add constraints for each $e_i \in E$

$$\sum_{j:e_i \in S_j} x_j \geq 1$$

In particular, this constraint says that for each $e_i$, there is one $x_j = 1$ meaning at least one $S_j$ chosen in the final set cover that contains it. The final linear program is given by:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \qquad i = 1, \ldots, n \\ & x_j \in \{0, 1\}, \quad j = 1, \ldots, m \end{aligned}$$

### 1.4.2 Relaxing the LP

Now let's sit back and relax the integer constraints. Replace $x_j \in \{0, 1\}$ with $x_j \geq 0$ to derive the following.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n \\ & x_j \geq 0, \quad j = 1, \ldots, m \end{aligned}$$

This is indeed a relaxation because all feasible solution for integer linear programming is also a feasible solution for this linear programming. We always have $Z^*_{LP} \leq Z^*_{ILP} = OPT$ where $Z^*_{LP}$ is the optimal value of the linear programming and $Z^*_{ILP}$ is the value for integer linear programming because the integer solution is always feasible for the relaxed program.

Our relaxed LP will return a vector of real numbers. We now need to round the real number solutions to integer solutions in order to return the combinatorial solution.

### 1.4.3 Deterministic Rounding

One way of rounding a linear programming is to set a certain threshold and accept if each variable if it's larger than that threshold. We use a rounding scheme as following:

**Deterministic Rounding Algorithm**
Given the solution to linear programming $x_j^*$, and $f = \max_i |\{j : e_i \in S_j\}|$. Initialize $I = \varnothing$

1. For each $j$, If $x_j^* \geq \frac{1}{f}$, Update $I \leftarrow I \cup \{S_j\}$.
2. return $I$.

$f$ is defined as the maximum number of sets that covers a single element for each element. The rounding procedure includes subset $S_j$ in our solution if and only if $x_j^* \geq 1/f$. To analyze this algorithm we need to complete two tasks: (1) demonstrate that this rounding scheme actually returns a valid set cover and (2) show that this set cover is a reasonable approximation. Let's first show that $I$ is a set cover.

**Lemma 1.4.** *$I$ is a set cover.*

*Proof.* for each element $e_i$. it's covered if in ILP solution $\sum_{j:e_i \in S_j} x_j \geq 1$. We know $x^*$ is a feasible solution, therefore we have $\sum_{j:e_i \in S_j} x_j^* \geq 1$. Because by definition, there must be $f_i \leq f$ terms in the summation, so there's at least a set in the summation that is included in the solution. □

And now we analyze the approximation ratio for this algorithm.

**Theorem 1.5.** *The rounding algorithm is an $f$-approximation algorithm for the set cover problem*

*Proof.* Observe that we can analyze the cost of the algorithm's set cover $I$ as follows.

$$\sum_{j \in I} w_j \leq \sum_{j=1}^{m} w_j \cdot (f \cdot x_j^*)$$

This inequality follows as $S_j$ is included in $I$ only if $x_j^* \geq 1/f$ or $f \cdot x_j^* \geq 1$. However, this implies

$$\sum_{j=1}^{m} w_j \cdot (f \cdot x_j^*) = f \sum_{j=1}^{m} w_j x_j^* = f \cdot Z_{LP}^* \leq f \cdot \text{OPT}$$

Notice here that we use the fact that $\sum_{j=1}^{m} w_j x_j^*$ is the objective value of LP optimal solution. As mentioned previously, this lower bounds the integer optimal solution since the integer solution is always LP feasible. This gives us a way to compare the algorithm's solution and the optimal solution allowing us to conclude that this is an $f$-approximation. □

An $f$-approximation algorithm is dependent on the input. For certain cases, this can provide for us a tighter bound. For example, this immediately provides a 2-approximation for vertex cover. Vertex cover is a special

case of set cover where you want to find the minimum set of vertices such that each edge is adjacent to at least one vertex in the set. Since each edge can only be adjacent to two vertices, the algorithm above immediately admits an $f = 2$ approximation.

A rounding of LP can also be used to produce a tighter approximation ratio after getting the solution for each instance. We know from the analysis that we have approximation ratio $\alpha \leq f$. However, it could be that the approximation ratio for an instance is much smaller than $f$. Comparing the solution we have with the LP relaxation could sometimes give us a better approximation ratio for each instance.

### 1.4.4 Randomized Rounding

Another variant of rounding relies on treating the solution probabilistically. Suppose we construct a random cover of $E$ by choosing each $S_j$ with probability $x_j^*$. If we let $X_j$ denote the random variable that $S_j$ is included in the sampled set cover, then the expected cost of the random cover is exactly LP solution.

$$\mathbf{E}\left[\sum_{j=1}^{m} w_j X_j\right] = \sum_{j=1}^{m} w_j \Pr[X_j = 1] = \sum_{j=1}^{m} w_j x_j^* = Z_{LP}^*$$

Probability that a given element $e_i$ not covered:

$$\Pr[e_i \text{ not covered}] = \prod_{j:e_i \in S_j} (1 - x_j^*) \leq \prod_{j:e_i \in S_j} e^{-x_j^*} = e^{-\sum_{j:e_i \in S_j} x_j^*} \leq e^{-1}$$

This means that with constant probability, the random set cover will fail to cover all elements! We want the probability that the set cover fails to be much smaller – say $\frac{1}{n^c}$ – so that we have by a union bound:

$$\Pr[\exists \text{ uncovered element}] \leq \sum_{i=1}^{n} \Pr[e_i \text{not covered}] \leq \frac{1}{n^{c-1}}$$

The way to deal with that is to give it a boost by repeating the process many times. Take an unfair coin that flips heads with probability $x_j^*$ and flip it $c \ln n$ times. Include set $S_j$ if at least one of the coin toss is heads.

$$\Pr[e_i \text{ not covered}] = \prod_{j:e_i \in S_j} (1 - x_j^*)^{c \ln n} \leq \prod_{j:e_i \in S_j} e^{-x_j^*(c \ln n)} = e^{-c \ln n \sum_{j:e_i \in S_j} x_j^*} \leq \frac{1}{n^c}$$

The following summarizes our *randomized* rounding algorithm.

**Randomized Rounding Algorithm**
Given the solution to the linear program $x_j^*$, initialize $I = \varnothing$. For each $j = 0, 1, ..., m$:

1. Flip a biased coin, which lands on head with probability $x_j^*$, $c \ln n$ times.
2. If at least one of the coin tosses is head, Update $I \leftarrow I \cup \{S_j\}$
3. return $I$.

Since our algorithm is random, it is possible that it does not return a reasonable approximation let alone a valid set cover. Instead, we can show that it returns a valid set cover that reasonably approximates the optimal *with high probability.*

**Theorem 1.6.** *This algorithm returns an $O(\ln n)$-approximation of set cover with high probability.*

*Proof.* $\Pr[X_j = 1] \leq (c \ln n) x_j^*$

$$\begin{aligned} E\left[\sum_{j=1}^{m} w_j X_j\right] &= \sum_{j=1}^{m} w_j \Pr[X_j = 1] \\ &\leq c \ln n \sum_{j=1}^{m} w_j x_j^* = c \ln n Z_{LP}^* \end{aligned} \tag{1.1}$$

need to provide E given set cover is provided: using conditional expectation. $E[A] = E[A|B]\Pr[B] + E[A|\neg B]\Pr[\neg B]$. Then for $n \geq 2$ and $c \geq 2$:

$$\begin{aligned} E\left[\sum_{j=1}^{m} w_j X_j \middle| F\right] &= \frac{1}{\Pr[F]}\left(E\left[\sum_{j=1}^{m} w_j X_j\right] - E\left[\sum_{j=1}^{m} w_j X_j \middle| \neg F\right]\Pr[\neg F]\right) \\ &\leq \frac{1}{\Pr[F]} E\left[\sum_{j=1}^{m} w_j X_j\right] \\ &\leq \frac{(c \ln n) Z_{LP}^*}{1 - \frac{1}{n^{c-1}}} \\ &\leq 2c \ln n Z_{LP}^* \end{aligned}$$

□

### 1.4.5 Dual Solution

In CS170, one may have encountered the concept of LP duality. Call a given maximization LP, the primal LP. We can compute the *dual linear program* of the primal which provides the tightest lower bound programs objective value. That is to say it abides by the following property.

**Theorem 1.7** (Weak Duality)**.** *If $x$ is a feasible solution to the primal LP with a maximization objective, and $y$ a feasible solution to the dual LP with a minimization objective, then the value of dual objective function $\leq$ the value of primal objective function.*

We can take the dual of the set cover LP that we derived and consider rounding a solution from there. The dual for set cover can be written as

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} y_i \\ \text{subject to} \quad & \sum_{i:e_i \in S_j} y_i \leq w_j \quad j = 1, \ldots, m \\ & y_i \geq 0 \qquad\qquad\quad i = 1, \ldots, n \end{aligned} \tag{1.2}$$

We can interpret this program in the following manner. To lower bound the primal objective value, we imagine a show owner pricing elements in each set $S_j$. In the primal setting, selecting $S_j$ for the cover means

a buyer can always pay $w_j$ for all elements in $S_j$. To maximize the profit the seller can make by selling each item separately, he can assign non-negative prices $y_i \geq 0$ (hence the dual constraint). However, prices must be assigned such that for each set, the sum of prices is at most $w_j$ as the buyer could always pay that much for the set. One can always derive the dual such that the following property holds.

We will round the dual solution in the following manner.

**Dual Rounding Algorithm**
Given a solution to the dual linear programming $y^*$, initialize $I = \varnothing$ and do the following.

1. Return $I$ given by
$$I = \left\{ S_j \subseteq E : \sum_{i:e_i \in S_j} y_i = w_j \right\}$$

Let us first verify that a solution $I'$ returned by this algorithm is a valid set cover.

**Lemma 1.8.** *The collection $\{S_j\}_{j \in I'}$ is a set cover.*

*Proof.* For sake of contradiction, suppose that there exists some elements $e_k$ uncovered. Then for each subset $S_j$ containing $e_k$, it must for all the sets j containing $e_k$:

$$\sum_{i:e_i \in S_j} y_i^* < w_j$$

as the algorithm only includes sets for which the constraint $\sum_{i:e_i \in S_j} y_i^* \leq w_j$ is achieved with equality. Let $\epsilon$ be the minimum difference between the price of all elements in $S_j$ and $w_j$. Construct

$$y_k' = y_k^* + \epsilon$$

And set $y_i' = y_i^*$ for all $i \neq k$. We immediately have $\sum_{i:e_i \in S_j} y_i^* \leq w_j$ for all sets containing $e_k$ meaning $y'$ is feasible for LP 1.2. However, $y'$ achieves a larger objective value as

$$\sum_{i=1}^{n} y_i' = \epsilon + \sum_{i=1}^{n} y_i^* > \sum_{i=1}^{n} y_i^*$$

This is not possible as $y^*$ is assumed to be the optimal solution for dual LP 1.2 – contradiction. □

Finally, we conclude that the dual rounding algorithm is an $f$-approximation.

**Theorem 1.9.** *This algorithm is an f-approximation algorithm for the set cover problem.*

*Proof.* Let us begin by writing out the cost of the dual rounded solution by recalling that $w_j = \sum_{i:e_i \in S_j} y_i^*$.

$$\sum_{j \in I'} w_j = \sum_{j \in I'} \sum_{i:e_i \in S_j} y_i^* = \sum_{i=1}^{n} |\{ j \in I' : e_i \in S_j \}| \cdot y_i^*$$

The last equality holds as for a fixed $i$, the double sum sums $y_i^*$ exactly the number of times $e_i$ appears in any $S_j$ in the cover. We then have the following:

$$\sum_{i=1}^{n} |\{j \in I' : e_i \in S_j\}| \cdot y_i^* \leq \sum_{i=1}^{n} f_i y_i^* \leq f \sum_{i=1}^{n} y_i^* \leq f \cdot OPT$$

allowing us to conclude the dual rounding algorithm returns an $f$-approximation. □

### 1.4.6 Primal-Dual Method

It remains expensive to solve large linear programs. The fastest known algorithm for linear programming can find the an approximate optimum in the time it takes to multiply two matrices [**?**]. At time of writing, matrix multiplication takes time $O(n^\omega)$ for $\omega \approx 2.37$ [2]. Special purpose rounding algorithm are often much faster. Take the rounding algorithm for dual programming in the previous subsection as an example, instead of trying to solve dual LP optimally, we are just constructing its feasible solutions. Therefore we can design a similar algorithm that achieves this purpose by repeatedly increasing dual variables until the constraints become tight. Such a technique is called *primal-dual* rounding and will be explored in future notes.

## 1.5 Conclusion

This note explored many different ways to approximate the set cover problem. Throughout the semester, we will be looking into these techniques in much more depth and will be introducing some other related techniques.

## References

[1] Cohen, M. B., Lee, Y. T., & Song, Z. (2018). "Solving linear programs in the current matrix multiplication time." in *arXiv preprint arXiv:1810.07896*.

[2] Williams, V. V. (2012). "Multiplying matrices faster than coppersmith-winograd." In *Symposium on the Theory of Computing* (Vol. 12, pp. 887-898).