CS 583: Approximation Algorithms: Covering Problems[*]

Chandra Chekuri

January 19, 2018

1 Introduction

Packing and Covering problems together capture many important problems in combinatorial optimization. We will consider covering problems in these notes. Two canonical covering problems are Vertex Cover and its generalization Set Cover. They play an important role in the study of approximation algorithms.

A *vertex cover* of a graph $G = (V, E)$ is a set $S \subseteq V$ such that for each edge $e \in E$, at least one end point of e is in S. Note that we are picking vertices to *cover* the edges. In the Vertex Cover problem, our goal is to find a smallest vertex cover of G. In the *weighted* version of the problem, a weight function $w : V \to \mathcal{R}^+$ is given, and our goal is to find a minimum weight vertex cover of G. The unweighted version of the problem is also known as Cardinality Vertex Cover.

In the Set Cover problem the input is a set $\mathcal{U}$ of n elements, and a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of m subsets of $\mathcal{U}$ such that $\bigcup_i S_i = \mathcal{U}$. Our goal in the Set Cover problem is to select as few subsets as possible from $\mathcal{S}$ such that their union covers $\mathcal{U}$. In the weighted version each set S_i has a non-negative weight w_i the goal is to find a set cover of minimim weight. Closely related to the Set Cover problem is the Maximum Coverage problem. In this problem the input is again $\mathcal{U}$ and $\mathcal{S}$ but we are also given an integer $k \leq m$. The goal is to select k subsets from $\mathcal{S}$ such that their union has the maximum cardinality. Note that Set Cover is a minimization problem while Maximum Coverage is a maximization problem. Set Cover is essentially equivalent to the Hitting Set problem. In Hitting Set the input is $\mathcal{U}$ and $\mathcal{S}$ but the goal is to pick the smallest number of elements of $\mathcal{U}$ that cover the given sets in $\mathcal{S}$. In other words we are seeking a set cover in the dual set system.

Set Cover is an important problem because it and its special cases not only arise as an explicit problems but also because many *implicit* covering problems can be cast as special cases. Consider the well known MST problem in graphs. One way to phrase MST is the following: given an edge-weighted graph $G = (V, E)$ find a minimum cost subset of the edges that *cover* all the cuts of G. This may appear to be a strange way of looking at the MST problem but this view is useful as we will see later.

Covering problems have the feature that a superset of a feasible solution is also feasible. More abstractly one can cast covering problems as the following. We are given a finite ground set V (vertices in a graph or sets in a set system) and a family of feasible solutions $\mathcal{I} \subseteq 2^V$ where $\mathcal{I}$ is upward closed; by this we mean that if $A \in \mathcal{I}$ and $A \subset B$ then $B \in \mathcal{I}$. The goal is to find the smallest cardinality set A in $\mathcal{I}$. In the weighted case V has weights and the goal is to find a minimum weight set in $\mathcal{I}$.

[*]Based on previous scribed lecture notes of Abul Hassan Samee and Lewis Tseng.

2 Approximating Set Cover and Maximum Coverage via Greedy

In this section we consider the unweighted version of Set Cover.

2.1 Greedy approximation

Both Set Cover and Maximum Coverage are known to be **NP**-Hard. A natural greedy approximation algorithm for these problems is as follows.

```
Greedy Cover ($\mathcal{U}, \mathcal{S}$):
1: repeat
2:     pick the set that covers the maximum number of uncovered elements
3:     mark elements in the chosen set as covered
4: until done
```

In case of Set Cover, the algorithm Greedy Cover is *done* in line 4 when all the elements in set $\mathcal{U}$ have been covered. And in case of Maximum Coverage, the algorithm is *done* when exactly k subsets have been selected from $\mathcal{S}$.

2.2 Analysis of Greedy Cover

Theorem 1 Greedy Cover *is a* $1 - (1 - 1/k)^k \geq (1 - \frac{1}{e}) \simeq 0.632$ *approximation for* Maximum Coverage, *and a* $(\ln n + 1)$ *approximation for* Set Cover.

The following theorem due to Feige [3] implies that Greedy Cover is essentially the best possible in terms of the approximation ratio that it guarantees in Theorem 1.

Theorem 2 *Unless* $\boldsymbol{NP} \subseteq \boldsymbol{DTIME}(n^{O(\log\log n)})$, *there is no* $(1 - o(1)) \ln n$ *approximation for* Set Cover. *Unless* **P=NP**, *for any fixed* $\epsilon > 0$, *there is no* $(1 - \frac{1}{e} - \epsilon)$ *approximation for* Maximum Coverage.

We proceed towards the proof of Theorem 1 by providing analysis of Greedy Cover separately for Set Cover and Maximum Coverage. Let OPT denote the value of an optimal solution to the Maximum Coverage problem. Let x_i denote the number of *new* elements covered by Greedy Cover in the i-th set that it picks. Also, let $y_i = \sum_{j=1}^{i} x_i$ be the number of elements covered after i iterations, and $z_i = OPT - y_i$. Note that, according to our notations, $y_0 = 0$, y_k is the number of elements chosen by Greedy Cover, and $z_0 = OPT$.

Analysis for Maximum Coverage

We have the following lemma for algorithm Greedy Cover when applied on Maximum Coverage.

Lemma 3 Greedy Cover *is a* $1 - (1 - 1/k)^k \geq 1 - \frac{1}{e}$ *approximation for* Maximum Coverage.

We first prove the following two claims.

Claim 4 *For* $i \geq 0$, $x_{i+1} \geq \frac{z_i}{k}$.

Proof: At each step, Greedy Cover selects the subset S_j whose inclusion covers the maximum number of uncovered elements. Since the optimal solution uses k sets to cover OPT elements, some set must cover at least $1/k$ fraction of the at least z_i remaining uncovered elements from OPT. Hence, $x_{i+1} \geq \frac{z_i}{k}$. □

Note that the Greey algorithm covers $x_1 + x_2 + \ldots + x_k$ elements. To analyze the worst-case we want to make this sum as small as possible given the preceding claim. Heuristically (which one can formalize) one can argue that choosing $x_{i+1} = z_i/k$ minimizes the sum. Using this one can argue that the sum is at least $(1-(1-1/k)^k)OPT$.

Claim 5 *For $i \geq 0$, $z_i \leq (1-\frac{1}{k})^i \cdot OPT$*

Proof: The claim is trivially true for $i = 0$ since $z_0 = OPT$. We assume inductively that $z_i \leq (1-\frac{1}{k})^i \cdot OPT$. Then

$$\begin{aligned} z_{i+1} &= z_i - x_{i+1} \\ &\leq z_i(1-\frac{1}{k}) \quad \text{[using Claim 4]} \\ &\leq (1-\frac{1}{k})^{i+1} \cdot OPT. \end{aligned}$$

□

Proof of Lemma 3. It follows from Claim 5 that $z_k \leq (1-\frac{1}{k})^k \cdot OPT \leq \frac{OPT}{e}$. Hence, $y_k = OPT - z_k \geq (1-\frac{1}{e}) \cdot OPT$. □

Analysis for Set Cover

We have the following lemma.

Lemma 6 Greedy Cover *is a* $(\ln n + 1)$ *approximation for* Set Cover.

Let k^* denote the value of an optimal solution to the Set Cover problem. Then an optimal solution to the Maximum Coverage problem for $k = k^*$ would cover all the n elements in set $\mathcal{U}$, and from our previous analysis $z_{k^*} \leq \frac{n}{e}$. Therefore, at most $\frac{n}{e}$ elements would remain uncovered after the first k^* steps of Greedy Cover. Similarly, after $2 \cdot k^*$ steps of Greedy Cover, at most $\frac{n}{e^2}$ elements would remain uncovered. This easy intuition convinces us that Greedy Cover is a $(\ln n + 1)$ approximation for the Set Cover problem. A more succinct proof is given below.

Proof of Lemma 6. Since $z_i \leq (1-\frac{1}{k^*})^i \cdot n$, after $t = k^* \ln \frac{n}{k^*}$ steps, $z_t \leq k^*$. Thus, after t steps, k^* elements are left to be covered. Since Greedy Cover picks at least one element in each step, it covers all the elements after picking at most $k^* \ln \frac{n}{k^*} + k^* \leq k^*(\ln n + 1)$ sets. □

A useful special case of Set Cover is when all sets are "small". Does the approximation bound for Greedy improve? We can prove the following corollary via Lemma 6.

Corollary 7 *If $|S_i| \leq d$, then* Greedy Cover *is a* $(\ln d + 1)$ *approximation for* Set Cover.

Proof: If each set has at most d elements then we have that $k^* \geq \frac{n}{d}$ and hence $\ln \frac{n}{k^*} \leq \ln d$. Then the claim follows from Lemma 6. □

Proof of Theorem 1. The claims follow directly from Lemma 3 and 6. □

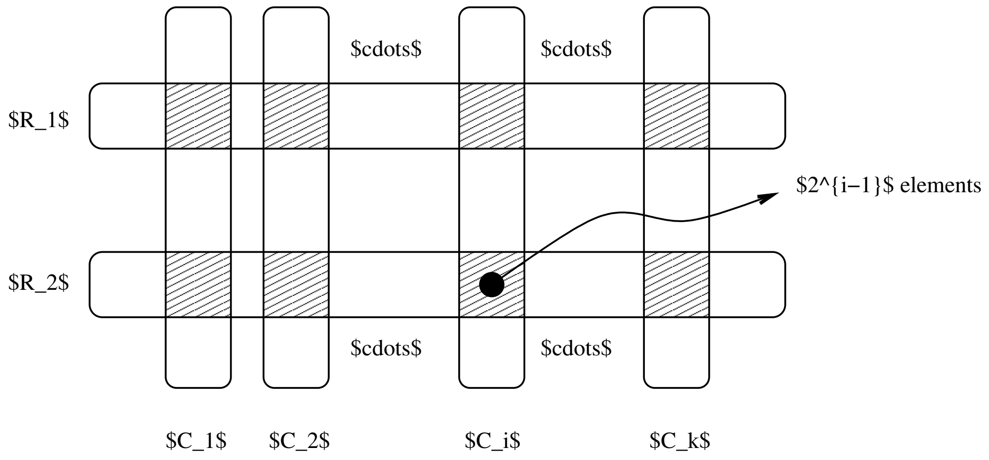

Figure 1: A near-tight example for Greedy Cover when applied on Set Cover

A near-tight example for Greedy Cover when applied on Set Cover

Let us consider a set $\mathcal{U}$ of n elements along with a collection $\mathcal{S}$ of $k+2$ subsets $\{R_1, R_2, C_1, C_2, \ldots, C_k\}$ of $\mathcal{U}$. Let us also assume that $|C_i| = 2^i$ and $|R_1 \cap C_i| = |R_2 \cap C_i| = 2^{i-1}$ $(1 \leq i \leq k)$, as illustrated in Fig. 1.

Clearly, the optimal solution consists of only two sets, i.e., R_1 and R_2. Hence, $OPT = 2$. However, Greedy Cover will pick each of the remaining k sets, namely $C_k, C_{k-1}, \ldots, C_1$. Since $n = 2 \cdot \sum_{i=0}^{k-1} 2^i = 2 \cdot (2^k - 1)$, we get $k \approx \Omega(\log_2 n)$. One can construct tighter examples with more involved analysis.

Exercise: Consider the weighted version of the Set Cover problem where a weight function $w : \mathcal{S} \rightarrow \mathcal{R}^+$ is given, and we want to select a collection $\mathcal{S}'$ of subsets from $\mathcal{S}$ such that $\cup_{X \in \mathcal{S}'} X = \mathcal{U}$, and $\sum_{X \in \mathcal{S}'} w(X)$ is the minimum. One can generalize the greedy heuristic in the natural fashion where in each iteration the algorithm picks the set that maximizes the ratio of the number of elements to its weight. Adapt the unweighted analysis to obtain an $O(\ln n)$ approximation for the weighted version.

2.3 Dominating Set

A *dominating set* in a graph $G = (V, E)$ is a set $S \subseteq V$ such that for each $u \in V$, either $u \in S$, or some neighbor v of u is in S. In the Dominating Set problem, our goal is to find a smallest dominating set of G.

A natural greedy algorithm for this problem is to iteratively choose a vertex with the highest degree. It can be shown that this heuristic gives a $(\ln n + 1)$, or more accurately, a $(\ln(\Delta + 1) + 1)$ approximation for the Dominating Set problem. Here Δ is the maximum degree of the given graph.

Exercises:

1. Show that Dominating Set is a special case of Set Cover.

2. Prove the approximation guarantees of the greedy heuristic for Dominating Set.

3. Show that SET COVER can be reduced in an approximation preserving fashion to DOMINATING SET. More formally, show that if DOMINATING SET has an $\alpha(n)$-approximation where n is the number of vertices in the given instance then SET COVER has an $(1 - o(1))\alpha(n)$-approximation.

3 VERTEX COVER

We have already seen that the VERTEX COVER problem is a special case of the SET COVER problem. It follows that the greedy algorithms gives an $O(\ln \Delta + 1)$ approximation for the unweighted versions of the VERTEX COVER problem. One can wonder wheter the Greedy algorith has a better worst-case for VERTEX COVER than the analysis suggests. Unfortunately the answer is negative and there are examples where the algorithm outputs a solution with $\Omega(\log n \cdot OPT)$ vertices.

We sketch the construction. Consider a bipartite graph $G = (U, V, E)$ where $U = \{u_1, u_2, \ldots, u_h\}$. V is partitioned into $S_1, S_2, \ldots, S_h$ where S_i has $\lfloor h/i \rfloor$ vertices. Each vertex v in S_i is connected to exactly i *distinct vertices* of U; thus, any vertex u_j is incident to at most one edge from S_i. It can be seen that the degree of each vertex $u_j \in U$ is roughly h. Clearly U is a vertex cover of G since the graph is bipartite. Convince yourself that the Greedy algorithm will pick all of V starting with the lone vertex in S_h (one may need to break ties to make this happen but the example can be easily perturbed to make this unnecessary). We have $n = \Theta(h \log h)$ and $OPT \le h$ and Greedy outputs a solution of size $\Omega(h \log h)$.

3.1 Better (constant) approximation for VERTEX COVER

CARDINALITY VERTEX COVER : The following is a 2-approximation algorithm for the CARDINALITY VERTEX COVER problem.

```
MATCHING-VC (G):
1: S ← ∅
2: Compute a maximal matching M in G
3: for each edge (u, v) ∈ M do
4:         add both u and v to S
5: Output S
```

Theorem 8 MATCHING-VC *is a 2-approximation algorithm.*

The proof of Theorem 8 follows from two simple claims.

Claim 9 *Let OPT be the size of the vertex cover in an optimal solution. Then $OPT \ge |M|$.*

Proof: Since the optimal vertex cover must contain at least one end vertex of every edge in M, $OPT \ge |M|$. □

Claim 10 *Let $S(M) = \{u, v | (u, v) \in M\}$. Then $S(M)$ is a vertex cover.*

Proof: If $S(M)$ is not a vertex cover, then there must be an edge $e \in E$ such that neither of its endpoints are in M. But then e can be included in M, which contradicts the maximality of M. □

Proof of Theorem 8. Since $S(M)$ is a vertex cover, Claim 9 implies that $|S(M)| = 2 \cdot |M| \le 2 \cdot OPT$. □

WEIGHTED VERTEX COVER: The matching based heuristic does not generalize in a straight forward fashion to the weighted case but 2-approximation algorithms for the WEIGHTED VERTEX COVER problem can be designed based on LP rounding.

3.2 SET COVER with small frequencies

VERTEX COVER is an instance of SET COVER where each element in $\mathcal{U}$ is in at most two sets (in fact, each element was in exactly two sets). This special case of the SET COVER problem has given us a 2-approximation algorithm. What would be the case if every element was contained in at most three sets? More generally, given an instance of SET COVER, for each $e \in \mathcal{U}$, let $f(e)$ denote the number of sets containing e. Let $f = \max_e f(e)$, which we call the *maximum frequency.*

Exercise: Give an f-approximation for SET COVER, where f is the maximum frequency of an element. *Hint:* Follow the approach used for VERTEX COVER .

4 Vertex Cover via LP

Let $G = (V, E)$ be an undirected graph with arc weights $w : V \to R^+$. We can formulate VERTEX COVER as an integer linear programming problem as follows. For each vertex v we have a variable x_v. We interpret the variable as follows: if $x_v = 1$ if v is chosen to be included in a vertex cover, otherwise $x_v = 0$. With this interprtation we can easily see that the minimum weight vertex cover can be formulated as the following integer linear program.

$$\begin{array}{ll} \min & \sum_{v \in V} w_v x_v \\ \text{subject to} & \\ x_u + x_v \geq 1 & \forall e = (u, v) \in E \\ x_v \in \{0, 1\} & \forall v \in V \end{array}$$

However, solving the preceding integer linear program is NP-Hard since it would solve VERTEX COVER exactly. Therefore we use Linear Programming (LP) to approximate the optimal solution, OPT(I), for the integer program. First, we can relax the constraint $x_v \in \{0, 1\}$ to $x_v \in [0, 1]$. It can be further simplified to $x_v \geq 0$, $\forall v \in V$.

Thus, a linear programming formulation for Vertex Cover is:

$$\begin{array}{ll} \min & \sum_{v \in V} w_v x_v \\ \text{subject to} & \\ x_u + x_v \geq 1 & \forall e = (u, v) \in E \\ x_v \geq 0 & \end{array}$$

We now use the following algorithm:

```
VERTEX COVER VIA LP:
Solve LP to obtain an optimal fractional solution $x^*$
Let $S = \{v \mid x^*_v \geq \frac{1}{2}\}$
Output $S$
```

Then the following claims are true:

Claim 11 *S is a vertex cover.*

Proof: Consider any edge, $e = (u, v)$. By feasibility of x^*, $x^*_u + x^*_v \geq 1$, and thus either $x^*_u \geq \frac{1}{2}$ or $x^*_v \geq \frac{1}{2}$. Therefore, at least one of u and v will be in S. □

Claim 12 $w(S) \leq 2\mathrm{OPT}_{LP}(I)$.

Proof: $\mathrm{OPT}_{LP}(I) = \sum_v w_v x^*_v \geq \frac{1}{2} \sum_{v \in S} w_v = \frac{1}{2} w(S)$ □

Therefore, $\mathrm{OPT}_{LP}(I) \geq \frac{\mathrm{OPT}(I)}{2}$ for all instances I.

Note: For minimization problems: $\mathrm{OPT}_{LP}(I) \leq \mathrm{OPT}(I)$, where $\mathrm{OPT}_{LP}(I)$ is the optimal solution found by LP; for maximization problems, $\mathrm{OPT}_{LP}(I) \geq \mathrm{OPT}(I)$.

Integrality Gap

We introduce the notion of *integrality gap* to show the best approximation guarantee we can acquire by using the LP optimum as a lower bound.

Definition: For a minimization problem Π, the integrality gap for a linear programming relaxation/formulation LP for Π is $\sup_{I \in \pi} \frac{\mathrm{OPT}(I)}{\mathrm{OPT}_{LP}(I)}$.

That is, the integrality gap is the worst case ratio, over all instances I of Π, of the integral optimal value and the fractional optimal value. Note that different linear programming formulations for the same problem may have different integrality gaps.

Claims 11 and 12 show that the integrality gap of the Vertex Cover LP formulation above is at most 2.

Question: Is this bound tight for the Vertex Cover LP?

Consider the following example: Take a complete graph, K_n, with n vertices, and each vertex has $w_v = 1$. It is clear that we have to choose $n - 1$ vertices to cover all the edges. Thus, $\mathrm{OPT}(K_n) = n - 1$. However, $x_v = \frac{1}{2}$ for each v is a feasible solution to the LP, which has a total weight of $\frac{n}{2}$. So gap is $2 - \frac{1}{n}$, which tends to 2 as $n \to \infty$.

Exercise: The vertex cover problem can be solved optimally in polynomial time in bipartite graphs. In fact the LP is integral. Prove this via the maxflow-mincut theorem and the integrality of flows when capacities are integral.

Other Results on Vertex Cover

1. The current best approximation ratio for Vertex Cover is $2 - \Theta(\frac{1}{\sqrt{\log n}})$ [5].

2. It is known that unless $P = NP$ there is α-approximation for Vertex Cover for $\alpha < 1.36$ [2]. Under a stronger hypothesis called the Unique Games Conjecture it is known that there is no $2 - \epsilon$ approximation for any fixed $\epsilon > 0$ [6].

3. There is a polynomial time approximation scheme (PTAS), that is a $(1 + \epsilon)$-approximation for any fixed $\epsilon > 0$, for planar graphs. This follows from a general approach due to Baker [1]. The theorem extends to more general classes of graphs.

5 Set Cover via LP

The input to the Set Cover problem consists of a finite set $U = \{1, 2, ..., n\}$, and m subsets of U, $S_1, S_2, ..., S_n$. Each set S_j has a non-negative weigh w_j and the goal is to find the minimum weight collection of sets which cover all elements in U (in other words their union is U).

A linear programming relaxation for Set Cover is:

$$\begin{array}{lrcll} \min & \sum_j w_j x_j & & & \\ \text{subject to} & & & & \\ & \sum_{j:i\in S_j} x_j & \geq & 1 & \forall i \in \{1,2,...,n\} \\ & x_j & \geq & 0 & 1 \leq j \leq m \end{array}$$

And its dual is:

$$\begin{array}{lrcll} \max & \sum_{i=1}^{n} y_i & & & \\ \text{subject to} & & & & \\ & \sum_{i\in S_j} y_i & \leq & w_j & \forall i \in \{1,2,...,n\} \\ & y_i & \geq & 0 & \forall i \in 1,2,...,n \end{array}$$

We give several algorithms for Set Cover based on this primal/dual pair LPs.

5.1 Deterministic Rounding

```
SET COVER VIA LP:
Solve LP to obtain an optimal solution $x^*$, which contains fractional numbers.
Let $P = \{i \mid x_i^* \geq \frac{1}{f}\}$, where $f$ is the maximum number of sets that contain any element
Output $\{S_j \mid j \in P\}$
```

Note that the above algorithm, even when specialized to VERTEX COVER is different from the one we saw earlier. It includes all sets which have a strictly positive value in an *optimum* solution the LP.

Let x^* be an optimal solution to the primal LP, y^* be an optimum solution to the dual, and let $P = \{j \mid x_j^* > 0\}$. First, note that by strong duality, $\sum_j w_j x_j^* = \sum_i y_i^*$. Second, by complementary slackness if $x_j^* > 0$ then the corresponding dual constraint is tight, that is $\sum_{i\in S_j} y_i^* = w_j$.

Claim 13 *The output of the algorithm is a feasible set cover for the given instance.*

Proof: Exercise. □

Claim 14 $\sum_{j\in P} w_j \leq f \sum_j w_j x_j^* = \text{OPT}_{LP}$.

Proof:

$$\sum_{j\in P} w_j = \sum_{j:x_j^*>0} (w_j) = \sum_{j:x_j^*>0} \left(\sum_{i\in S_j} y_i^*\right) = \sum_i y_i^* \left(\sum_{j:i\in S_j, x_j^*>0} 1\right) \leq f \sum_i y_i^* \leq f\text{OPT}_{LP}(I).$$

. $\square$

Notice that the the second equality is due to complementary slackness conditions (if $x_j > 0$, the corresponding dual constraint is tight), the penultimate inequality uses the definition of f, and the last inequality follows from weak duality (a feasible solution for the dual problem is a lower bound on the optimal primal solution).

Therefore we have that the algorithm outputs a cover of weight at most $f\mathrm{OPT}_{LP}$. We note that f can be as large as n in which case the bound given by the algorithm is quite weak. In fact, it is not construct examples that demonstrate the tightness of the analysis.

Remark: The analysis cruically uses the fact that x^* is an optimal solution. On the other hand the algorithm for VERTEX COVER is more robust and works with any feasible solution x. It is easy to generalize the earlier rounding for VERTEX COVER to obtain an f-approximation. The point of the above rounding is to illustrate the utility of complementary slackness.

5.2 Randomized Rounding

Now we describe a different rounding that yields an approximation bound that does not depend on f.

```
SOLVING SET COVER VIA RANDOMIZED ROUNDING:
$A = \emptyset$, and let $x^*$ be an optimal solution to the LP.
for $k = 1$ to $2\ln n$ do
    pick each $S_j$ independently with probability $x_j^*$
    if $S_j$ is picked, $A = A \cup \{j\}$
end for
Output the sets with indices in $A$
```

Claim 15 *$Pr[i$ is not covered in an iteration$] = \prod_{j:i\in S_j}(1 - x_j^*) \le \frac{1}{e}$.*

Intuition: We know that $\sum_{j:i\in S_j} x_j^* \ge 1$. Subject to this constraint, if and want to minimize the probability, we can let x_j^* equal to each other, then the probability $= (1-\frac{1}{k})^k$, where $x_j^* = 1/k$.

Proof: $Pr[i$ is not covered in an iteration$] = \prod_{j:i\in S_j}(1 - x_j^*) \le \prod_{j:i\in S_j} e^{-x_j^*} \le e^{-\sum_{j:i\in S_j} x_j^*} \le \frac{1}{e}$. $\square$

We then obtain the following corollaries:

Corollary: $Pr[i$ is not covered at the end of the algorithm$] \le e^{-2\log n} \le \frac{1}{n^2}$.

Corollary: $Pr[$all elements are covered, after the algorithm stops$] \ge 1 - \frac{1}{n}$. The above follows from the union bound. The probability that i is not covered is at most $1/n^2$, hence the probability that there is some i that is not covered is at most $n \cdot 1/n^2 \le 1/n$.

Let C_t = cost of sets picked in iteration t, then $E[C_t] = \sum_{j=1}^m w_j x_j^*$, where $E[X]$ denotes the expectation of a random variable X. Then, let $C = \sum_{t=1}^{2\ln n} C_t$; we have $E[C] = \sum_{t=1}^{2\ln n} E[C_t] \le 2\ln n\mathrm{OPT}_{LP}$. We know that $Pr[C > 2E[C]] \le \frac{1}{2}$ by Markov's inequality, so we have $Pr[C \le 4\ln n\mathrm{OPT}_{LP}] \ge \frac{1}{2}$. Therefore, $Pr[C \le 4\ln n\mathrm{OPT}_{LP}$ and all items are covered$] \ge \frac{1}{2} - \frac{1}{n}$. Thus, the randomized rounding algorithm, with probability close to $1/2$ succeeds in giving a feasible solution of cost $O(\log n)\mathrm{OPT}_{LP}$. Note that we can check whether the solution satisfies the desired properties (feasibility and cost) and repeat the algorithm if it does not.

1. We can check if solution after rounding satisfies the desired properties, such as all elements are covered, or cost at most $2c \log n \mathrm{OPT}_{LP}$. If not, repeat rounding. Expected number of iterations to succeed is a constant.

2. We can also use Chernoff bounds (large deviation bounds) to show that a single rounding succeeds with high probability (probability at least $1 - \frac{1}{poly(n)}$).

3. The algorithm can be *derandomized.* Derandomization is a technique of removing randomness or using as little randomness as possible. There are many derandomization techniques, such as the method of conditional expectation, discrepancy theory, and expander graphs.

4. After a few rounds, select the cheapest set that covers each uncovered element. This has low expected cost. This algorithm ensures feasibility but guarantees cost only in the expected sense. We will see a variant on the homework.

Other Results related to Set Cover

1. Unless $P = NP$, there is no $c \log n$ approximation for some fixed c [8].

2. Unless NP $\subseteq DTIME(n^{O(\log \log n)})$, there is no $(1 - o(1)) \ln n$-approximation [3].

3. Unless $P = NP$, there is no $(1 - \frac{1}{e} + \varepsilon)$-approximation for max-coverage for any fixed $\varepsilon > 0$ [3].

5.3 Dual-fitting

In this section, we introduce the technique of dual-fitting for the analysis of approximation algorithms. At a high-level the approach is the following:

1. Construct a feasible solution to the dual LP.

2. Show that the cost of the solution returned by the algorithm can be bounded in terms of the value of the dual solution.

Note that the algorithm itself need not be LP based. Here, we use Set Cover as an example. Please refer to the previous section for the primal and dual LP formulations of Set Cover.

We can interpret the dual as follows: Think of y_i as how much element i is willing to pay to be covered; the dual maximizes the total payment, subject to the constraint that for each set, the total payment of elements in that set is at most the cost of the set.

The greedy algorithm for weighted Set Cover is as follows:

```
GREEDY SET COVER:
$Covered = \emptyset$;
$A = \emptyset$;
While $Covered \neq U$ do
    $j \leftarrow \arg\min_k \left(\frac{w_k}{|S_k \cap \ Uncovered|}\right)$;
    $Covered = Covered \cup S_j$;
    $A = A \cup \{j\}$.
end while;
Output sets in $A$ as cover
```

Theorem 16 GREEDY SET COVER *picks a solution of cost* $\leq H_d \cdot \mathrm{OPT}_{LP}$*, where* d *is the maximum set size, i.e.,* $d = \max_j |S_j|$.

To prove this, we can augment the algorithm a little bit:

```
AUGMENTED GREEDY ALGORITHM OF WEIGHTED SET COVER:
$Covered = \emptyset$;
while $Covered \neq U$ do
    $j \leftarrow \arg\min_k (\frac{w_k}{|S_k \cap\ Uncovered|})$;
    if $i$ is uncovered and $i \in S_j$, set $p_i = \frac{w_j}{|S_j \cap\ Uncovered|}$;
    $Covered = Covered \cup S_j$;
    $A = A \cup \{j\}$.
end while;
Output sets in $A$ as cover
```

It is easy to see that the algorithm outputs a set cover.

Claim 17 $\sum_{j \in A} w_j = \sum_i p_i$.

Proof: Consider when j is added to A. Let $S'_j \subseteq S_j$ be the elements that are uncovered before j is added. For each $i \in S'_j$ the algorithm sets $p_i = w_j / |S'_j|$. Hence, $\sum_{i \in S'_j} p_i = w_j$. Moreover, it is easy to see that the sets S'_j, $j \in A$ are disjoint and together partition U. Therefore,

$$\sum_{j \in A} w_j = \sum_{j \in A} \sum_{i \in S'_j} p_i = \sum_{i \in U} p_i.$$

□

For each i, let $y'_i = \frac{1}{H_d} p_i$.

Claim 18 y' *is a feasible solution for the dual LP.*

Suppose the claim is true, then the cost of GREEDY SET COVER's solution $= \sum_i p_i = H_d \sum_i y'_i \leq H_d \mathrm{OPT}_{LP}$. The last step is because any feasible solution for the dual problem is a lower bound on the value of the primal LP (weak duality).

Now, we prove the claim. Let S_j be an arbitrary set, and let $|S_j| = t \leq d$. Let $S_j = \{i_1, i_2, ..., i_t\}$, where we the elements are ordered such that i_1 is covered by Greedy no-later than i_2, and i_2 is covered no later than i_3 and so on.

Claim 19 *For* $1 \leq h \leq t$, $p_{i_h} \leq \frac{w_j}{t-h+1}$.

Proof: Let $S_{j'}$ be the set that covers i_h in Greedy. When Greedy picked $S_{j'}$ the elements $i_h, i_{h+1}, \ldots, i_t$ from S_j were uncovered and hence Greedy could have picked S_j as well. This implies that the density of $S_{j'}$ when it was picked was no more than $\frac{w_j}{t-h+1}$. Therefore p_{i_h} which is set to the density of $S_{j'}$ is at most $\frac{w_j}{t-h+1}$. □

From the above claim, we have

$$\sum_{1 \leq h \leq t} p_{i_h} \leq \sum_{1 \leq h \leq t} \frac{w_j}{t-h+1} = w_j H_t \leq w_j H_d.$$

Thus, the setting of y'_i to be p_i scaled down by a factor of H_d gives a feasible solution.

5.4 Greedy for implicit instances of Set Cover

Set Cover and the greedy heuristic are quite useful in applications because many instances are *implicit*, nevertheless, the algorithm and the analysis applies. That is, the universe $\mathcal{U}$ of elements and the collection $\mathcal{S}$ of subsets of $\mathcal{U}$ are not restricted to be finite or explicitly enumerated in the Set Cover problem. For instance, a problem could require covering a finite set of points in the plane using disks of unit radius. There is an infinite set of such disks, but the greedy approximation algorithm can still be applied. For such implicit instances, the greedy algorithm can be used if we have access to an *oracle*, which, at each iteration, selects a set having the optimal density. However, an oracle may not always be capable of selecting an optimal set. In such cases, it may have to make the selections *approximately.* We call an oracle an α*-approximate oracle* if, at each iteration, it selects a set S such that $density(S) \geq \alpha \cdot Optimal\ Density$, for some $\alpha > 1$.

Exercise: Prove that the approximation guarantee of greedy approximation with an α-approximate oracle would be $\alpha(\ln n + 1)$ for Set Cover, and $(1 - \frac{1}{e^\alpha})$ for Maximum Coverage.

We will see several examples of implicit use of the greedy analysis in the course.

6 Extensions of Set Cover and Covering Integer Programs (CIPs)

There are several extensions of Set Cover that are interesting and useful such as set multicover. We refer to the reader to the relevant chapters in the two reference books. Here we refer to a general problem called Covering Integer Programs (CIPs for short). The goal is to solve the following *integer* program where $A \in \mathbb{R}_+^{n \times m}$ is a *non-negative* matrix. We can assume without loss of generality that w and b are also non-negative.

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} w_j x_j \\
\text{subject to} \quad & \\
Ax &\geq b \\
x_j &\leq d_j \qquad 1 \leq j \leq m \\
x_j &\geq 0 \qquad 1 \leq j \leq m \\
x_j &\in \mathbb{Z} \qquad 1 \leq j \leq m
\end{aligned}$$

$Ax \geq b$ model covering constraints and $x_j \leq d_j$ models multiplicity constraints. Note that Set Cover is a special case where A is simply the incidence matrix of the sets and elements (the columns correspond to sets and the rows to elements) and $d_j = 1$ for all j. What are CIPs modeling? It is a generalization of Set Cover . To see this, assume, without loss of generality, that A, b are integer matrices. For each element corresponding to row i the quantity b_i corresponds to the requirement of how many times i needs to be covered. A_{ij} corresponds to the number of times set S_j covers element i. d_j is an upper bound on the number of copies of set S_j that are allowed to be picked.

One can apply the Greedy algorithm to the above problem and the standard analysis shows that the approximation ratio obtained is $O(\log B)$ where $B = \sum_i b_i$ (assuming that they are integers). Even though this is reasonable we would prefer a strongly polynomial bound. In fact there are instances where B is exponential in n and the worst-case approximation ratio can be poor. One can obtain an $O(\log n)$ approximation but it is quite technical. The natural LP relaxation of the above integer program has very poor integrality gap in constrat to the case of SET COVER . One needs to *strengthen* the LP relaxation via what are known as *knapsack cover inequalities.* We refer the reader to the paper of Kolliopoulos and Young [7].

Open Problem: Is there a simple combinatorial algorithm for CIPs that yields an $O(\log^c n)$-approximation for some $c \geq 1$?

7 Submodularity

SET COVER turns out to be a special case of a more general problem called SUBMODULAR SET COVER and the greedy algorithm and analysis applies in this more generality as well. Submodularity is a fundamental notion with many applications in combinatorial optimization and else where. Here we take the opportunity to provide some definitions and a few results.

Given a finite set E, a set function $f : 2^E \rightarrow \mathcal{R}$ that assigns a value to each subset of E is submodular iff

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad \forall A, B \subseteq E.$$

Alternatively, f is a submodular functions iff

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B) \quad \forall A \subset B, i \in E \setminus B.$$

The second characterization shows that submodularity is based on *decreasing marginal utility* property in the discrete setting. Adding element i to a set A will help at least as much as adding it to to a (larger) set $B \supset A$. It is common to use $A + i$ to denote $A \cup \{i\}$ and $A - i$ to denote $A \setminus \{i\}$.

Exercise: Prove that the two characterizations of submodular functions are equivalent.

Many application of submodular functions are when f is a non-negative function though there are several important applications when f can be negative. A submodular function $f(\cdot)$ is *monotone* if $f(A + i) \geq f(A)$ for all $i \in E$ and $A \subseteq E$. Typically one assumes that f is normalized by which we mean that $f(\emptyset) = 0$; this can always be done by shifting the function by $f(\emptyset)$. f is *symmetric* if $f(A) = f(E \setminus A)$ for all $A \subseteq E$. Submodular set functions arise in a large number of fields including combinatorial optimization, probability, and geometry. Examples include rank function of a matroid, the sizes of cutsets in a directed or undirected graph, the probability that a subset of events do not occur simultaneously, entropy of random variables, etc. In the following we show that the SET COVER and MAXIMUM COVERAGE problems can be easily formulated in terms of submodular set functions.

Exercise. Suppose we are given a universe $\mathcal{U}$ and a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of $\mathcal{U}$. Now if we take $N = \{1, 2, \ldots, m\}$, $f : 2^N \rightarrow \mathcal{R}^+$, and define $f(A) = |\cup_{i \in A} S_i|$ for $A \subseteq E$, then show that the function f is a monotone submodular set function.

7.1 SUBMODULAR SET COVER

When formulated in terms of submodular set functions, the SET COVER problem is the following. Given a monotone submodular function f (whose value would be computed by an oracle) on

$N = \{1, 2, \ldots, m\}$, find the smallest set $S \subseteq N$ such that $f(S) = f(N)$. Our previous greedy approximation can be applied to this formulation as follows.

```
GREEDY SUBMODULAR $(f, N)$:
1: $S \leftarrow \emptyset$
2: while $f(S) \neq f(N)$
3:     find $i$ to maximize $f(S + i) - f(S)$
4:     $S \leftarrow S \cup \{i\}$
```

Not so easy Exercises:

1. Can you prove that the greedy algorithm is a $1 + \ln(f(N))$ approximation for SUBMODULAR SET COVER?
2. Can you prove that the greedy algorithm is a $1 + \ln(\max_i f(i))$ approximation for SUBMODULAR SET COVER.

The above results were first obtained by Wolsey [10].

7.2 SUBMODULAR MAXIMUM COVERAGE

By formulating the MAXIMUM COVERAGE problem in terms of submodular functions, we seek to maximize $f(S)$ such that $|S| \leq k$. We can apply algorithm GREEDY SUBMODULAR for this problem by changing the condition in line 2 to be: **while** $|S| \leq k$.

Exercise: Prove that greedy gives a $(1 - 1/e)$-approximation for SUBMODULAR MAXIMUM COVERAGE problem when f is monotone and non-negative. *Hint:* Generalize the main claim that we used for max coverage.

The above and many related results were shown in the influential papers of Fisher, Nemhauser and Wolsey [9, 4].

References

[1] Brenda S Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[2] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

[3] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[4] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions II. *Polyhedral combinatorics*, pages 73–87, 1978.

[5] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 5(4):41, 2009.

[6] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[7] Stavros G Kolliopoulos and Neal E Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71(4):495–505, 2005.

[8] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

[9] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[10] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.