CSIS0351/CSIS8601: Randomized Algorithms
Lecture 3: Set Cover: Randomized Rounding
Lecturer: Hubert Chan
Date: 15 Sept 2011

These lecture notes are supplementary materials for the lectures. They are by no means substitutes for attending lectures or replacement for your own notes!

1 Set Cover

Consider the following problems.

1. Suppose there is a set $\mathcal{U}$ of n households that wish to receive TV signals. Moreover, there are m possible locations to install signal amplifiers such that at location j, the amplifier can cover households in some subset $S_j \subseteq \mathcal{U}$. What is the minimum number of amplifiers that are sufficient to cover every household?

2. Suppose a buyer wants to purchase a complete set $\mathcal{U}$ of n cards. The cards are sold as m possible bundles such that for bundle j, the price is $c(j)$ and it includes a subset $S_j \subseteq \mathcal{U}$ of cards. What is the minimum cost the buyer needs to pay such that he can collect one complete set of cards?

These problems can be formulated in terms of the Set Cover Problem.

Definition 1.1 (Set Cover (Unweighted Version)) *Consider the following problem.*

Input: *A set $\mathcal{U}$ of n elements; a collection $\{S_j : j \in [m]\}$ of m subsets of $\mathcal{U}$.*

Output: *A feasible subset $C \subseteq [m]$ such that the sub-collection $\{S_j : j \in C\}$ covers $\mathcal{U}$, i.e., $\cup_{j \in C} S_j = \mathcal{U}$. We wish to minimize the cost $|C|$ of the solution, which is the number of subsets used to cover $\mathcal{U}$.*

Definition 1.2 (Set Cover (Weighted Version)) *Consider the following problem.*

Input: *A set $\mathcal{U}$ of n elements; a collection $\{S_j : j \in [m]\}$ of m subsets of $\mathcal{U}$ such that S_j has cost $c(j)$.*

Output: *A feasible subset $C \subseteq [m]$ such that the sub-collection $\{S_j : j \in C\}$ covers $\mathcal{U}$, i.e., $\cup_{j \in C} S_j = \mathcal{U}$. We wish to minimize the cost $\sum_{j \in C} c(j)$ of the solution, which is the total cost of the subsets used to cover $\mathcal{U}$.*

Observe that the unweighted version is a special case of the weighted version, in which every subset has cost 1. We first concentrate on the unweighted version of the problem. The techniques can be applied to the weighted version in a similar fashion.

Definition 1.3 (Approximation Ratio) *An approximation algorithm for a cost minimization problem is said to have approximation ratio $\rho \geq 1$ if for any instance of the problem, the algorithm returns a feasible solution with cost at most $\rho \cdot \mathsf{OPT}$, where OPT is the optimal cost for that instance.*

Unfortunately, a result of Feige's [Fei98] states that there is no efficient algorithm for set cover with approximation ratio less than $\ln n$, assuming $P \neq NP$.

Theorem 1.4 (Hardness of Set Cover [Fei98]) *For all $0 < \epsilon < 1$, it is NP-hard to approximate the Set Cover Problem with approximation ratio $(1-\epsilon)\ln n$.*

In view of the hardness result, we aim for approximation ratio $\Theta(\log n)$ for the Set Cover Problem.

2 Approximation Algorithm via Linear Programming

We first consider an equivalent reformulation of the problem. Suppose we are given an instance of the problem, i.e., we have a set $\mathcal{U}$ of n elements and a collection $\{S_j : j \in [m]\}$ of m subsets.

For each $j \in [m]$, we assign an indicator variable $x_j \in \{0,1\}$ such that if we want to use the subset S_j, we set $x_j := 1$ and otherwise $x_j := 0$. Then, the total number of subsets used is $\sum_{j\in[m]} x_j$.

However, observe that the variables x_j's have to satisfy some constraint. In particular, each element $g \in U$ needs to be covered by at least one subset. Hence, out of the subsets S_j's that contain g, at least one of the corresponding variables x_j's has to be 1. Therefore, we have the condition $\sum_{j:g\in S_j} x_j \geq 1$, for all $g \in \mathcal{U}$. We have the following integer linear program formulation for the given instance.

(1) Integer Linear Program Formulation

$$\begin{array}{rrlll} \mathsf{OPT} := & \min & \sum_{j\in[m]} x_j & & \\ & \forall g \in \mathcal{U}: & \sum_{j:g\in S_j} x_j & \geq & 1 \\ & \forall j \in [m]: & x_j & \in & \{0,1\} \end{array}$$

This is exactly the same problem as before, and hence it is still hard to solve. The hardness of the problem is actually because of the integer constraints $x_j \in \{0,1\}$, for all $j \in [m]$. If we relax the constraint to such that x_j can take any non-negative reals $x_j \geq 0$, we obtain a *linear relaxation* of the problem.

(2) Linear Program Relaxation

$$\begin{array}{rrlll} \mathsf{LP} := & \min & \sum_{j\in[m]} x_j & & \\ & \forall g \in \mathcal{U}: & \sum_{j:g\in S_j} x_j & \geq & 1 \\ & \forall j \in [m]: & x_j & \geq & 0 \end{array}$$

Here are some questions one might ask: Why do we relax the integer constraint? How does this affect the cost of the solution?

1. After relaxing the integer constraints $x_j \in \{0,1\}$ to $x_j \geq 0$, the linear program can be solved efficiently, i.e., non-negative values (possibly fractional) for x_j's can be found efficiently such that the objective function is minimized.

   A Note on LP Solver. The interested reader can refer to the `lp_solve` package in the following link: `http://lpsolve.sourceforge.net/5.5/`

2. Observe that every feasible solution for (1) is also feasible for (2). Hence, if OPT is the optimal cost for (1) and LP is the optimal cost for (2), we have $\mathsf{LP} \leq \mathsf{OPT}$. Note that if the minimum for (2) is attained only at some fractional solution, then $\mathsf{LP} < \mathsf{OPT}$.

3. We could be relax $x_j \in \{0, 1\}$ to $0 \leq x_j \leq 1$. However, $x_j \leq 1$ is unnecessary because since we only need to satisfy constraint of the type $\sum_{j:e \in S_j} x_j \geq 1$, there is no need to assign values greater than 1 to any variable.

Although we can solve the Linear Program Relaxation (2), the fractional solution does not solve our original problem. For instance, if $x_2 = \frac{1}{2}$, what does it mean to use half of the set S_2? However, the fractional solution gives us a clue on how to pick the subsets. In particular, we can try the following randomized approach: if $x_j = p$ for some $p \in [0, 1]$, then we can include S_j in our solution with probability p.

3 Randomized Rounding

We consider the method of randomized rounding, which was first used by Raghavan and Thompson in [RT87].

3.1 First Attempt

Given an instance of set cover, we consider the Linear Program Relaxation (2), and use an LP solver to obtain an optimal solution $\{x_j\}$ to the (2). We perform the following rounding procedure.

Simple Rounding Procedure

1. For each $j \in [m]$, independently pick S_j with probability x_j, i.e., include j in C with probability x_j. (Observe that for an optimal LP solution, we must have $0 \leq x_j \leq 1$.)

2. Return C as a candidate solution.

We first analyze the expected size of C. Since each subset S_j is picked with probability x_j, it follows that the expected size of C is exactly $\sum_{j \in [m]} x_j = \mathsf{LP}$, which is at most OPT. This is good news, as the expected size of C is at most the optimal (feasible) solution of the original problem.

However, the problem is that C might not be feasible. In fact, most probably there is some element that is not covered by the subsets indicated by C. Let us first try to obtain an upper bound on the probability that a particular element g is not covered.

Observe that element g is not covered exactly when all the subsets S_j containing g are not picked, and this happens with probability $\prod_{j:g \in S_j}(1 - x_j) \leq \prod_{j:g \in S_j} \exp(-x_j) = \exp(-\sum_{j:g \in S_j} x_j) \leq \exp(-1)$. The last inequality holds because the x_j's form a feasible solution to the LP relaxation and hence $\sum_{j:g \in S_j} x_j \geq 1$.

3.2 Second Attempt

We know that in the Simple Rounding Procedure, for each element g, the probability that it is not covered is at most $\frac{1}{e}$. Suppose we repeat the Simple Rounding Procedure T times independently.

In particular, we do the following.

1. For $t = 1, 2, \ldots, T$, do:
   Run the Simple Randomized Procedure to obtain cover C_t.
2. Return $\widehat{C} := \cup_{t=1}^{T} C_t$.

In other words, we return the cover indexed by $\widehat{C}$, which is the union of all the C_t's returned in the repeated runs.

Note that a subset can be repeated several times. However, we still have the upper bound $E[|\widehat{C}|] \leq T \cdot \mathsf{LP} \leq T \cdot \mathsf{OPT}$. Let B_1 be the event that $|\widehat{C}| \geq 4T \cdot \mathsf{OPT}$. Then, by Markov's inequality, $Pr[B_1] \leq \frac{1}{4}$.

The probability that an element g is not covered by $\widehat{C}$ is at most $(\frac{1}{e})^T$. This is at most $\frac{1}{4n}$ if we pick $T := \ln 4n$. Let B_2 be the event that there exists some element which is not covered by $\widehat{C}$. Then, $Pr[B_2] \leq \frac{1}{4}$, by the union bound over all elements in $\mathcal{U}$.

It follows that $Pr[B_1 \cup B_2] \leq \frac{1}{2}$. Hence, with probability at least $\frac{1}{2}$, the subsets $\{S_j : j \in \widehat{C}\}$ cover all elements in $\mathcal{U}$, and the number of subsets used is at most $4T \cdot \mathsf{OPT} = 4 \ln 4n \cdot OPT$.

Currently, the failure probability is at most $\frac{1}{2}$. Do you know how to decrease the failure probability to arbitrary $\delta > 0$?

4 Greedy Algorithm

We can also compute a set cover by the following greedy algorithm.

Greedy Algorithm. Initialize $C := \emptyset$ and $V := \mathcal{U}$. (The variable C holds the indices of the subset chosen so far, and V is the set of covered elements in $\mathcal{U}$.) While V is non-empty, find a subset S_j such that $|V \cap S_j|$ is maximized; set $C := C \cup \{j\}$ and $V := V \setminus S_j$. Return C when V is empty.

Theorem 4.1 *Suppose $B := \max_j |S_j|$. Then, the greedy algorithm gives $\ln B + 1$ approximation ratio.*

Dual Variables. In order to analyze the greedy algorithm, we use some auxiliary variables. Recall that the set cover problem can be formulated as an integer program.

$$\begin{array}{rrcl} \mathsf{OPT} := & \min & \sum_{j \in [m]} x_j & \\ & \forall g \in \mathcal{U} : & \sum_{j : g \in S_j} x_j & \geq \quad 1 \\ & \forall j \in [m] : & x_j & \in \quad \{0, 1\} \end{array}$$

In the LP literature, the x_j's are known as the primal variables. For each inequality, we consider a variable y_g, for each $g \in \mathcal{U}$. The y_g's are known as dual variables that assist the analysis.

We consider the greedy algorithm again, but this time we assign values to the dual variables for book keeping. Suppose initially all variables are set to 0. When some subset S_j is selected by the algorithm, i.e. when we include j in C. At this point, for each $g \in S_j \cap V$, we set $y_g := \frac{1}{|S_j \cap V|}$.

Claim 4.2 *When the greedy algorithm terminates, the number of subsets chosen is* $\sum_{g\in\mathcal{U}} y_g$.

We next analyze the dual variables. For each integer n, we denote $H(n) := \sum_{r=1}^{n} \frac{1}{r} \leq \ln n + 1$.

Theorem 4.3 *Suppose* S_j *is some subset such that* $|S_j| = B_j$. *Then, when the greedy algorithm terminates,* $\sum_{g\in S_j} y_g \leq H(B_j)$.

Proof: Consider the elements in S_j. Initially, all of them are uncovered. Look at the moment when any element in S_j are being covered at the first time. This must be caused by the inclusion of some subset S' (which may or may not be S_j).

However, since S' is chosen, we know that it must contain at least as many uncovered elements as S_j, i.e., $|S' \cap V| \geq |S_j \cap V| = B_j$.

At this point, for all $g \in S_j \cap S'$, we assign $y_g := \frac{1}{|S'\cap V|} \leq \frac{1}{B_j}$.

Hence, if k elements in $S_j \cap S'$ are covered in this step, all those k corresponding variables y_g's will receive a value at most $\frac{1}{B_j}$. However, we use a looser upper bound $\sum_{g\in S_j\cap S'} y_g \leq \frac{1}{B_j} + \frac{1}{B_j-1} + \cdots + \frac{1}{B_j-k+1}$.

Observe now that S_j contains $B_j - k$ uncovered elements. Using the same argument, next time another element g from S_j is covered, the corresponding variable y_g will receive a value at most $\frac{1}{B_j-k}$.

Continuing the argument, it follows that eventually we have $\sum_{g\in S_j} y_g \leq \frac{1}{B_j} + \frac{1}{B_j-1} + \ldots + \frac{1}{2} + 1 = H(B_j)$. ■

Theorem 4.4 *Suppose* $B := \max_j |S_j| = \max_j B_j$. *Then,* $\sum_{g\in\mathcal{U}} y_g \leq H(B) \cdot \mathsf{OPT}$.

Proof: Consider an optimal solution given by x_j's such that $\sum_{j=1}^{m} x_j = \mathsf{OPT}$.

Observe that since the x_j's form a feasible cover, for all $g \in \mathcal{U}$, $\sum_{j:g\in S_j} x_j \geq 1$.

Hence, the greedy algorithm returns a cover with size

$\sum_{g\in\mathcal{U}} y_g \leq \sum_{g\in\mathcal{U}} y_g \sum_{j:g\in S_j} x_j = \sum_{g\in\mathcal{U}} \sum_{j:g\in S_j} y_g x_j$.

We change the order of summation, this equals

$\sum_{j=1}^{m} \sum_{g\in S_j} y_g x_j = \sum_{j=1}^{m} x_j \sum_{g\in S_j} y_g \leq \sum_{j=1}^{m} x_j \cdot H(B)$, where the last inequality follows from Theorem 4.3.

Since $\sum_{j=1}^{m} x_j = \mathsf{OPT}$, the result follows. ■

5 Homework Preview

1. **Better Approximation Ratio for Set Cover.** In the lecture notes, a randomized algorithm to achieve approximation ratio $4 \ln 4n$ is described. We show that if we increase the number of repetitions slightly, we can actually achieve a better ratio. We assume that the number of elements in $\mathcal{U}$ is large enough, say $n \geq 20$.

   (a) Suppose we repeat the Simple Rounding Procedure for $T := \ln n + \lambda \ln \ln n$ times, where $\lambda > 0$ is some number we determine later. Suppose $\widehat{C}$ is the index set of the cover

returned from the repeated runs, and we want approximation ratio ρ, for some $\rho > 1$ which we want as small as possible. Let B_1 be the event that $|\widehat{C}| \geq \rho \cdot \mathsf{OPT}$. Show that if you set $\rho := \ln n + 2\lambda \ln \ln n + 2$, then $Pr[B_1] \leq 1 - \frac{1}{\ln n}$.

(b) Let B_2 be the event that there is some element that is not covered by $\widehat{C}$. What value should λ take such that the $Pr[B_2] \leq \frac{1}{(\ln n)^2}$?

(c) What is the approximation ratio ρ (in terms of only n) of the resulting algorithm? Can you give an upper bound on the failure probability? Failure means either there is some element in $\mathcal{U}$ that is not covered or the cover $\widehat{C}$ is too large.

(d) Given any arbitrary $0 < \delta < 1$, how can you obtain the same approximation ratio, but with failure probability at most δ? How many times in total do you have to run the Simple Rounding Procedure?

References

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[RT87] Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.