4810-1183 Approximation and Online Algorithms
Lecture Note 4: Approximation Algorithm for Vertex Cover Problem

Introduction

On last week, we have introduced you an approximation algorithm for the knapsack problem. The algorithm looks very close to a greedy algorithm, an algorithmic paradigm where we will pick items in order of some "good" function until we cannot pick items anymore. However, it is not exactly a greedy algorithm. At the end of the algorithm, we choose between two sets of strawberries. Actually, many approximation algorithms are devised based on that paradigm. We usually get an approximation algorithm by slightly modifying a greedy algorithm.

The greedy-algorithm-based is one of the two most common schemes for approximation algorithm, and we will study the other common schemes today. The scheme is called as deterministic rounding.

Vertex Cover

Let recall our discussion on social networks in Lecture Note 2. We have a set of persons V and a set of friendships between two persons $E \subseteq \{\{u,v\}: u,v \in V\}$. Suppose that we are a spy from a particular country, and we want to track all information in a social network using social engineering. We will bribe a set of persons and make them reveal their communications with all friends. The bribing is very costly and we want to minimize the number of persons who we bribe.

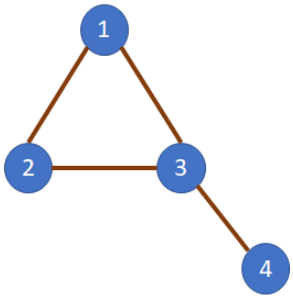
The problem in the previous paragraph can be formulized into the following optimization models:

Input: Set V, Set $E \subseteq \{\{u,v\}: u,v \in V\}$

Output: Set $S \subseteq$ V (set of persons who post the information)

Objective Function: Minimize |S|

Constraint: For all $\{u,v\} \in E$, $u \in S$ or $v \in S$



Consider the above social network, where $V = \{1,2,3,4\}$ and $E = \{\{1,2\},\{1,3\},\{2,3\},\{3,4\}\}$. We can spy all the information if we bribe 1 and 3, and it is not possible to cover all communications by just one person. The optimal solution is $S^* = \{1,3\}$ and the optimal value is $OPT = 2$.

Again, the optimization model is NP-hard. To solve the problem, let define new variables x_i for each $i \in V$. We have $x_i = 1$ when $i \in S$, and $x_i = 0$ otherwise. Suppose that $V = \{1, \dots, n\}$. It is straightforward to see that $x_1, \dots, x_n$ have an equivalent information with S. We can construct $x_1, \dots, x_n$ from S and we can construct S from $x_1, \dots, x_n$. Because of that, we will output $\mathbf{x} = [x_1, \dots, x_n]$ instead of S.

Then, we have $|S| = \sum_i x_i$, because, to $|S|$, i contributes $1 = x_i$ when $i \in S$ and $0 = x_i$ otherwise. When $\mathbf{c} = [1,1,\dots,1]^t$, we have $|S| = \mathbf{c}^t\mathbf{x}$.

Consider the objective function of our optimization model. For all $\{u, v\} \in E$, we want to have $u \in S$ or $v \in S$. We want to have $x_u = 1$ or $x_v = 1$. That is equivalent to have $x_u + x_v = 1$ or $x_u + x_v = 2$, and the conditions can be written as $x_u + x_v \geq 1$. In our previous example, we want to have the following conditions:

$$x_1 + x_2 \geq 1$$

$$x_1 + x_3 \geq 1$$

$$x_2 + x_3 \geq 1$$

$$x_3 + x_4 \geq 1$$

The conditions can be rewritten to the following form:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

When $\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, the condition is $\mathbf{Ax} \geq \mathbf{b}$. For the general case, we will have $\mathbf{A}$ be a $|E| \times |V|$ matrix where each row is corresponding to a friendship and each column is corresponding to a person. At a row that is corresponding to a friendship $\{u, v\}$, elements of $\mathbf{A}$ is 1 only at the column of persons u, v. The elements will be 0 at the other columns. The vector $\mathbf{b}$ is always be $[1, 1, \dots, 1]^t$.

By the notations above, we can reformulate our optimization model to the following:

Input:	Matrix $\mathbf{A}$, Vector $\mathbf{b}, \mathbf{c}$ *Assumption*: At each row of A, two of the elements are 1. The other elements are 0. All elements of $\mathbf{b}, \mathbf{c}$ are 1.
Output:	Vector $\mathbf{x}$ (all elements are 0 or 1)
Objective Function:	Minimize $\mathbf{c}^t\mathbf{x}$
Constraint:	$\mathbf{Ax} \geq \mathbf{b}$

The optimization model looks very close to linear programming. One might think that we can solve the optimization model using library like CPLEX®. There is only one difference: we require that x_i must be 0 or 1, but that difference makes us cannot use CPLEX® to solve the problem. Consider the previous example again. If we use CPLEX® to solve the problem without the conditions on x_i, the output could be $x_1 = x_2 = x_3 = x_4 = 0.5$. We know how to construct S from $\mathbf{x}$ when all x_i are 0 or 1, but we do not how to do that when x_i is not that 2 numbers.

Rounding

Now, let us consider the following optimization model, and call the problem as "fractional vertex cover".

Input: Matrix **A**, Vector $\mathbf{b}, \mathbf{c}$
Assumption: At each row of A, two of the elements are 1.
The other elements are 0. All elements of $\mathbf{b}, \mathbf{c}$ are 1.

Output: Vector $\mathbf{x}$ ($\mathbf{0} \leq \boldsymbol{x_i} \leq \mathbf{1}$ **for all** $\boldsymbol{i}$)

Objective Function: Minimize $\mathbf{c}^t\mathbf{x}$

Constraint: $\mathbf{Ax} \geq \mathbf{b}$

Instead of having $x_i \in \{0,1\}$, we have $x_i \in [0,1]$. Although a library for linear programming cannot solve our optimization models when the outputs must be within a discrete set, it theoretically works very well when the outputs must be within a single continuous set. Because of that, we can use a library for linear programming to solve the "fractional vertex cover".

Our algorithm is as follows:

1: Use a library for linear programming to solve the "fractional vertex cover". Suppose that the output is $\mathbf{x} = (x_1, \ldots, x_n)$.

2: For all i, let $x_i' = 1$ when $x_i \geq 0.5$, and $x_i = 0$ otherwise. Let $\mathbf{x}' = (x_1', \ldots, x_n')$.

3: Return x' as an answer of the vertex cover problem.

We will prove the correctness of the above algorithm in the previous theorem.

Theorem An output of the algorithm $\mathbf{x}'$ satisfies the constraint of vertex cover problem.

Proof: Because the vector **x** is a solution of the fractional vertex cover problem, we know that, for $\{u, v\} \in E$,

$$x_u + x_v \geq 1.$$

By the above inequality, we know that either x_u or x_v is not smaller than 0.5. If both of them are smaller, we will never have the summation larger than 1. By the rounding at line 2 of the algorithm, we know that either x_u' or x_v' are 1. For all $\{u, v\} \in E$, we have $x_u' + x_v' \geq 1$, which means that $\mathbf{x}'$ satisfies the constraint of the vertex cover problem.

■

Let SOL be the objective value of the output from our algorithm, denoted by $\mathbf{x}'$, let OPT be the optimal value, and $\mathbf{x}^*$ be the optimal solution.

We have the following theorem for the algorithm in the previous paragraph.

Theorem For any particular input, $SOL \leq 2 \cdot OPT$.

Proof: Let $\mathbf{x} = (x_1, \ldots, x_n)$ be the optimal solution for the fractional vertex cover problem. It is the vector in $[0,1]^n$ that minimizes $\mathbf{c}^t\mathbf{x}$ when $\mathbf{Ax} \geq \mathbf{b}$. On the other hand, $\mathbf{x}^*$ is the vector in $\{0,1\}^n$ that minimizes the $\mathbf{c}^t\mathbf{x}$ when $\mathbf{Ax} \geq \mathbf{b}$. Both of the vectors minimize the objective function with the same constraint, but **x** is

selected from a larger set. **x** has more chance to minimizes the objective function. The objective value of **x** is better (thus smaller) than $\mathbf{x}^*$, i.e. $\mathbf{c}^t\mathbf{x} \leq \mathbf{c}^t\mathbf{x}^*$.

Now, let consider the value of x_i'. By the "rounding" at Line 2 of the algorithm, we have $x_i' \leq 2 \cdot x_i$. Then,

$$SOL = \mathbf{c}^t\mathbf{x}' = \sum_i x_i' \leq 2 \cdot \sum_i x_i = 2 \cdot \mathbf{c}^t\mathbf{x} \leq 2 \cdot \mathbf{c}^t\mathbf{x}^* = 2 \cdot OPT.$$

■

Approximation Algorithm for Minimization Problems

In the previous lecture note, we work on the knapsack problem. There, we tried to "maximize" our happiness from strawberries. From the algorithm, we cannot have the maximum happiness as in an optimal solution, but we can prove $SOL \geq 0.5 \cdot OPT$. An algorithm that can attain the inequality is called 0.5-approximation algorithm.

In the vertex cover problem, we are trying to "minimize" our cost. We cannot have the minimum cost as in an optimal solution. The objective value of our output, SOL, must be larger than OPT. However, we prove that SOL is not larger than $2 \cdot OPT$. An algorithm that can attain that is called 2-approximation algorithm.

We have α-approximation algorithm for both maximization and minimization problems. However, $\alpha < 1$ for maximization problems and $\alpha > 1$ for minimization problems.

Network Monitoring Problem

Let consider a problem considered in [2]. Suppose again that we want to monitor all communications. Here, we do not have to bribe persons anymore because we are ISP. We have all communications passing our servers! However, it is quite costly to set up a monitoring tool at all servers, and we want to set up the tools at as least as servers possible. Our servers are connected so well that a communication will pass only of our two servers, ingress server (where the communication arrives) and forwarding server (where the communication is forwarded to other ISPs.

By the discussion at the previous paragraph, we have the following optimization model.

Input:	Number of servers n
	Number of communications m Ingress servers for each communication $I_1, \dots, I_m$ Forward servers for each communication $O_1, \dots, O_m$
Output:	Servers to set the monitoring tools $S \subseteq \{1, \dots, n\}$
Constraint:	For all communications j, $I_j \in S$ or $O_j \in S$
Objective Function:	Minimize $\lvert S \rvert$

The ingress and forward servers are equivalent in the above problem. Because of that, we can denote all information about a communication j by $\{I_j, O_j\}$, and restate the problem to the following way:

Input:	Set of servers $V = \{1, \dots, n\}$ Set of communications $E = \{\{I_1, O_1\}, \dots, \{I_m, O_m\}\}$
Output:	$S \subseteq V$
Constraint:	For all $\{I_j, O_j\} \in E$, $I_j \in S$ or $O_j \in S$
Objective Function:	Minimize $\lvert S \rvert$

The above optimization model is the vertex cover problem. We can use a 2-approximation algorithm to solve the problem.

References

[1] D. P. Williamson and D. Shmoys, "*The design of approximation algorithms*", Cambridge University Press, 2011. Chapter 4

[2] S. Agrawal, K. Naidu, and R. Rastogi, "*Diagnosing link-level anomalies using passive probes*", Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007), pages 1757-1765, 2007.