IOE 691: Approximation Algorithms — Date: 01/11/2017

Lecture Notes: K-center Hardness and Max-Coverage (Greedy)

Instructor: Viswanath Nagarajan — Scribe: Sentao Miao

1 Overview

In this lecture, we will talk about the hardness of a problem and some properties of problem reduction (a technique usually used to show some problem is NP-hard). Then using this technique, we will show that K-center problem is NP-hard for any approximation ratio less than 2.

Another topic is to develop an approximation algorithm with ratio $1-1/e$ for the maximum coverage problem.

2 Hardness of a Problem

Consider we have a problem P, it can be either a *decision problem* (if the output is yes or no) or an *optimization problem* (if the output is the maximum or minimum of the problem instance). However, we generally only consider the decision problem because decision and optimization problems are somehow “equivalent”. More specifically, think about the K-center problem. A decision problem of K-center is that given some bound B, whether there exists a solution whose objective is less or equal to B; the optimization problem of K-center is simply to minimize the objective. Obviously, if we know how to solve the optimization problem, we can immediately answer the decision problem. On the other hand, if we can answer any decision problem with any B, using a polynomial-time routine (e.g. binary search), we can find the optimal value of the optimization problem as well.

So consider a decision problem, how do we identify its hardness rigorously? There are two main classes of decision problems based on its hardness. The first class is called **P**. A problem Q is in **P** if given a decision instance (e.g. bound B in K-center problem), we can answer whether there exists a solution satisfying this instance (e.g. some solution with objective value $\leq B$ in K-center problem) in *polynomial time* of the input size. The other class is called **NP**. Loosely speaking, problem Q is in **NP** if given a decision instance *and a solution*, we can check whether this solution satisfies the instance in polynomial time of input size.

Hence, we can obviously see that if Q is in **P**, it is in **NP**. However, the other direction remains open till now and whether **P=NP** is one of the most famous problems in mathematics and computer science. Currently, it is widely assumed that **P$\neq$NP** and this will be the underlying assumption for us as well.

Till now, we have “defined” two classes of problems with different hardness. Then the next question is, given a problem P how can we prove its hardness in a rigorous way? This is achieved from the so-called *reduction* of problems.

$$Q \longrightarrow P$$

Definition 2.1 ***Reduction*** *from problem Q (of input size n) to problem P is a map from instances I of Q to instances J of P satisfying the following properties.*

1. *The reduction takes polynomial time in n.*
2. *I is a yes-instance if and only if J is a yes-instance.*

Then we have the important notion of **NP-hard**. That is, problem P is **NP-hard** if it can be reduced from all **NP** problems. Loosely speaking, if problem P is **NP-hard**, we believe that there is no polynomial time algorithm such that any decision instance can be checked (unless **P=NP**). Why? Because otherwise, from the definition of reduction and **NP-hard**, all **NP** problems can be solved in polynomial time, which shows **P=NP**, a statement which we assume to be wrong. Based on this argument, we immediately has the following observation.

Observation: Suppose we can reduce Q to P. If Q is **NP-hard**, P is also **NP-hard**.

Remark: The notion of **NP-hard** can be extended to approximate problems as well. That is, we can show for some problems that even computing an α approximation to the optimal value is **NP-hard** (for a suitable α).

2.1 Approximation hardness of K-center problem

Now we will show the hardness of K-center problem.

Theorem 2.1 *Assuming **P≠NP**, there is no polynomial time algorithm for K-center problem with approximation ratio < 2.*

Proof: We use the known fact that *dominating set* is an NP-hard problem, see (Garey and Johnson 2002). The dominating set problem Q is the following. Given an undirected graph $G = (V, E)$ and bound K, we want to decide whether there exists a subset S of V of size K such that every vertex $v \in V$ is either in S or a neighbor of some $s \in S$. See Figure 1 as an example.

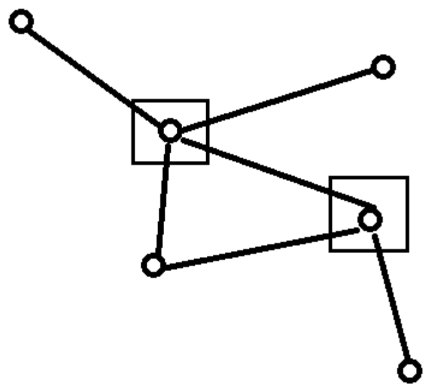

Figure 1: In this case, the nodes in rectangles form one dominating set.

Let P be the decision version of K-center problem. In particular, let V be the vertices of K-center problem, and the distance function is defined as

$$\begin{aligned} d(u,v) &= 1 \qquad \text{if } (u,v) \in E \\ d(u,v) &= 2 \qquad \text{otherwise} \end{aligned}$$

We can easily check it is a metric. Also, let K be the same as in Q. The decision is, whether the optimum OPT is smaller or equal to $B := 1$, where OPT is the solution of

$$\min_{S:|S|\le K} \max_{v\in V} \quad d(v,S)$$

which is an equivalent version of K-center optimization problem. So we create a mapping from instances of Q to P.

We can also check that this mapping satisfies the second property of reduction, i.e. yes-instance of dominating set Q maps to the instance of $OPT \le 1$; and no-instance to $OPT \ge 2$. At the same time, for any approximation ratio $\alpha < 2$, the instances of $OPT \le 1$ is one-to-one mapped to $OBJ \le \alpha$. These relationships are represented by Figure 2. Therefore, if a polynomial time algorithm exists for K-center with approximation ratio < 2, then we get a polynomial time algorithm exists for dominating set problem, contradicting the assumption **P≠NP**

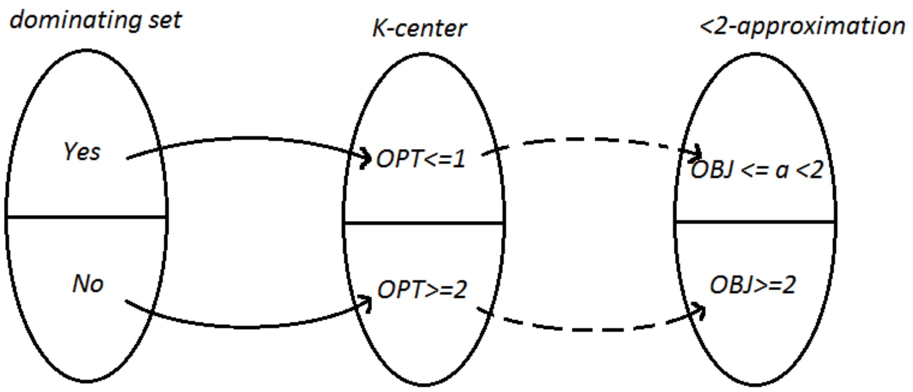

Figure 2: Reduction

■

3 Maximum Coverage Problem

Definition 3.1 (Maximum Coverage Problem) *Given a universe $V = \{e_1, ..., e_n\}$ of elements, a list of sets $\{S_i \subseteq V\}_{i=1}^m$ (possibly overlapping) and a bound K, the goal is to find K sets $S'_1, ..., S'_K$ such that $|\bigcup_{i=1}^K S'_i|$ is maximized.*

This is again an **NP-hard problem**, and we are going to give a greedy algorithm which has approximation ratio $1 - 1/e$.

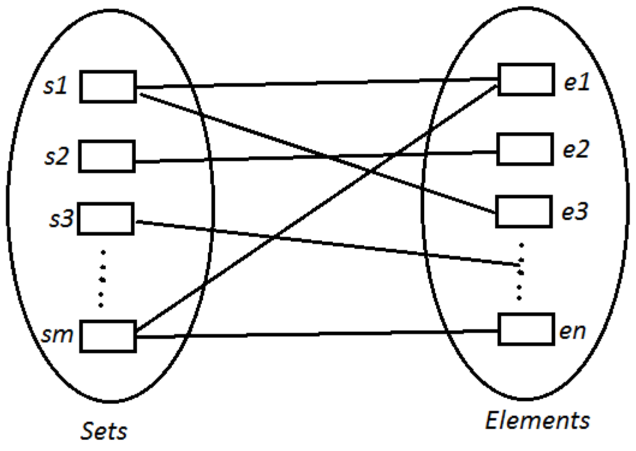

Figure 3: Maximum coverage problem

```
Data: $V$ : set of all elements; $S_1, ..., S_n$; $K$
Result: Approximate solution $A_1, \cdots, A_K$
$U = V$;
for $i = 1, ..., K$ do
    Let $A_i$ be one of the sets $S_1, ..., S_n$ which maximizes $|A_i \bigcap U|$;
    $U = U \backslash A_i$;
end
```

Algorithm 1: Greedy Algorithm for Maximum Coverage

The basic idea of the algorithm is to choose the set in each step which contains most of the uncovered elements. We have the following theorem.

Theorem 3.1 *Algorithm 1 is a* $(1 - 1/e)-$*approximation algorithm for maximum coverage.*

To prove this theorem, let us first prove the following two lemmas. Let C_i denote the set of elements covered by the end of iteration i in Algorithm 1.

Lemma 3.1 *For all* $i = 1, ..., K$, $|A_i \bigcap U| = |C_i| - |C_{i-1}| \geq \frac{OPT - |C_{i-1}|}{K}$

Proof: The number of elements covered in the optimal solution but not in the algorithm at the start of iteration i is $\geq OPT - |C_{i-1}|$. Let sets in the optimal solution be $S_1^*, ..., S_K^*$. Let $U = V \backslash C_{i-1}$.

Obviously, $\bigcup_{i=1}^{K}(S_i^* \bigcap U) = (\bigcup S_i^*) \backslash C_{i-1}$. This implies that

$$\sum_{i=1}^{K} |S_i^* \bigcap U| \geq |\bigcup_{i=1}^{K}(S_i^* \bigcap U)| \geq OPT - |C_{i-1}|$$

which further implies

$$\max_{i=1}^{K} |S_i^* \bigcap U| \geq \frac{OPT - |C_{i-1}|}{K}.$$

By definition, $|A_i \bigcap U| \geq \max_i |S_i^* \bigcap U|$ and we are done. ■

Lemma 3.2 $|C_i| \geq \frac{OPT}{K} \sum_{j=0}^{i-1}(1 - 1/K)^j$ *for all* $i = 1, ..., K$.

Proof: Prove by induction. The base case $i = 1$ is trivial as the first choice $A_1 = C_1$ has at least OPT/K elements by Lemma 3.1.

For inductive step, suppose i holds, and we want to prove that it holds for $i + 1$.

$$\begin{aligned}
|C_{i+1}| &\geq |C_i| + \frac{OPT - |C_i|}{K} \\
&= \frac{OPT}{K} + (1 - 1/K)|C_i| \\
&\geq \frac{OPT}{K} \sum_{j=0}^{i} (1 - 1/K)^j
\end{aligned}$$

The first inequality is by Lemma 3.1, and the last inequality is from inductive hypothesis. ■

Now we are ready to prove Theorem 3.1.

Proof: [of Theorem 3.1]

$$\begin{aligned}
|C_K| &\geq \frac{OPT}{K} \sum_{j=0}^{K-1} (1 - 1/K)^j \\
&= \frac{OPT}{K} \frac{1 - (1 - 1/K)^K}{1 - (1 - 1/K)} \\
&= OPT(1 - (1 - 1/K)^K) \\
&\geq (1 - 1/e)OPT
\end{aligned}$$

where the first inequality is from Lemma 3.2, and the last inequality is from the fact that $(1 - 1/K)^K \leq (e^{-1/K})^K$ which is obtained from $1 + x \leq e^x$ for all $x \in$ R. ■

References

Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.