Lecture 3: Greedy Algorithms

This is a very sketchy note that summarizes the material covered in class. Please do not expect any perfection out of it. If you have any question regarding the note, do not hesitate to contact me.

1 k-Center (WS Chapter 2.2)

A metric consists of a point set X and distance function $d : X \times X \to \mathbb{R}$ satisfying the following properties:

- $d(u,u) = 0$ for all $u \in X$
- $d(u,v) = d(v,u)$ for all $u, v \in X$
- $d(u,v) + d(v,w) \geq d(u,w)$ for any $u, v, w \in X$

For any subset $X' \subseteq X$, we write $d(u, X')$ to denote $\min_{v \in X'} d(u,v)$. Now the statement of the k center problem is as follows. Given a metric (X,d) and an integer k, our goal is to pick $S : |S| = k$ such that the maximum distance to S is minimized, i.e.

$$\min_{S \subseteq X : |S| = k} \max_{u \in X} d(u, S)$$

1.1 Algorithm

The greedy algorithm goes as follows. Starting from $S = \emptyset$, while $|S| < k$, we add the vertex in X whose distance to S is maximized, i.e. in each iteration, we pick vertex u such that $d(u,S)$ is maximized and add u to S.

The following lemma implies that the greedy algorithm is optimal.

Lemma 1.1. *Let R be the value of the optimal solution. When $|S| = k$, we have $d(S,u) \leq 2R$ for all $u \in X$.*

Proof. Let $S^* = \{v_1, \ldots, v_k\} \subseteq X, |S^*| = k$ be the optimal solution for the k-center instance. Divide points in X into k clusters $C_1, \ldots, C_k \subseteq S^*$ where C_i contain vertices in X that are closest to v_i, so we have that $d(v_i, u) \leq R$ for all $u \in C_i$. Now consider our greedy solution S. Suppose that $|S \cap C_i| = 1$ for all i, then we would be done: For any point $u \in C_i$, we have $d(u,S) \leq d(u,v_i) + d(v_i,S) \leq 2R$ (by triangle inequality). Otherwise, if $|S \cap C_i| > 1$ for some cluster i, then at some point of the greedy algorithm, there is no pair of vertices whose distance is more than $2R$. So in any case we can conclude that $d(S,u) \leq 2R$ for all $u \in X$. $\square$

2 Edge Disjoint Paths

We use the notation $[k]$ to denote the set of numbers $\{1, \ldots, k\}$. In the Edge Disjoint Path (EDP) problem, we are given graph $G = (V,E)$ and a collection of pairs $\{(s_i,t_i)\}_{i=1}^k$. Our objective is to find a subset $I \subseteq [k]$ and paths P_i for all $i \in I$ such that all paths $\{P_i\}_{i \in I}$ are edge disjoint.

We show that a simple greedy algorithm gives $\sqrt{|E|}$ approximation algorithm for EDP. In some cases, this algorithm gives the best known approximation ratio (e.g. planar graphs).

2.1 Algorithm

Define the notation, for each $i \in [k]$ and graph G', $sp(i, G')$ as a shortest path connecting s_i and t_i in G'. Initially, $I = \emptyset$ and $G' = G$. While there is still a path connecting some $i \in [k] \setminus I$ in G', we choose i^* as the index $i \in [k] \setminus I$ such that the length of $sp(i, G')$ is minimized; denote by $P_{i^*} = sp(i^*, G')$. Then we remove edges in P_{i^*} from G', add i^* to set I, and proceed to the next iteration.

2.2 Analysis

Denote by $m = |E(G)|$. Consider the collection of paths $\mathcal{P} = \{P_i\}_{i \in I}$ returned by the algorithm. Let I^* and $\mathcal{P}^*$ be the indices and the set of paths chosen by the optimal solution, where each index $i \in I^*$ is associated with a path $P_i^* \in \mathcal{P}^*$. We are going to compare the values of $|I|$ and $|I^*|$.

Claim 2.1. *For each $i \in I$, we either have $i \in I^*$, or there is some $j \neq i$ in I^* such that $P_i \cap P_j^* \neq \emptyset$ (otherwise, the greedy algorithm would not finish or the solution $\mathcal{P}^*$ would not be optimal). In such case, we say that P_j "conflicts" with P_i.*

[1]

We renumber the paths so that paths added into I are $I = \{1, \dots, |I|\}$, where P_i is added into the solution in iteration i. Notice that the greedy algorithm guarantees that $|P_i| \leq |P_j|$ for $i \leq j$. We partition the execution of the algorithm into two phases. In the first phase, all paths added into I have length less than $\sqrt{m}$ (thus referred to as *short*), and in the second phases, all paths are at least $\sqrt{m}$ in length. So we know that $P_1, \dots, P_s$ are short for some integer s, while $P_{s+1}, \dots$ are long. Let I_1^* be the indices $j \in I^*$ such that P_j conflicts with some paths in $P_1, \dots, P_s$.

Claim 2.2. $s \geq |I_1^*|/\sqrt{m}$.

[2]

Proof. This follows because each short path $i \in I$ conflicts with at most $\sqrt{m}$ paths in I_1^*. □

Now define $I_2^* = I^* \setminus I_1^*$. If $I_2^* = \emptyset$, we would be done, so we assume that $I_2^* \neq \emptyset$. This would also imply that $|I| \geq s + 1$. Notice that all paths in I_2^* must be long, because otherwise, P_{s+1} would have been short, but this also implies that I_2^* cannot contain more than $\sqrt{m}$ paths. All these imply that $|I| \geq s + 1 \geq |I^*|/\sqrt{m}$.

3 Minimum Makespan Scheduling (WS, Chapter 2.3)

There are n jobs with processing times $p_1, \dots, p_n$ respectively and m *identical* machines. Find an assignment of jobs to the machines so that the completion time (a.k.a. *makespan*) is minimized.

The problem is NP-hard even when there are only two machines.

3.1 An Algorithm

The greedy algorithm proceeds in n iterations as follows. In iteration i, try to schedule job i by adding it to the machine with least amount of work so far.

Let OPT denote the optimal makespan. We need the following lemma, whose proof is left as an exercise.

Lemma 3.1. $\mathsf{OPT} \geq \max\left\{\frac{\sum_j p_j}{m}, \max_j p_j\right\}$

[1] You should try to convince yourself that the claim is correct. I will not prove the claim formally.

[2] Again, you should try to fill in the detail of the proof by yourself.

Now we argue that the greedy algorithm is a 2 approximation algorithm. Consider machine i whose completion time is maximized. It is enough to bound the completion time of this machine. Let p_j be the last job that finishes on this machine which is scheduled to start at time t. It must be the case that $t \leq \frac{\sum_j p_j}{m}$ (because all machines must be full up to time t). Therefore we have that the makespan of this solution is at most $t + p_j \leq 2\mathsf{OPT}$. With a more careful analysis, one can show that this algorithm is in fact a $(2 - 1/m)$ approximation algorithm.

4 Multiway Cut

Given a graph $G = (V, E)$ with non-negative weights w and terminal set T, our goal is to compute a subset F of edges such that $(V, E \setminus F)$ disconnects every terminal pair, while minimizing the cost $w(F)$.

When $|T| = 2$, this problem is simply minimum s-t cut, so it can be solved in polynomial time. When $|T| = 3$, the problem already becomes NP-hard, so again we will design approximation algorithms for it.

4.1 An Algorithm

Denote by $T = \{t_1, \ldots, t_k\}$. Our algorithm computes, for each $i = 1, \ldots, k$, the minimum cut separating t_i and $T \setminus \{t_i\}$; call the set of resulting edges F_i. The final solution is defined as $F = \bigcup_{i=1}^{k} F_i$.

Lemma 4.1. $w(F) \leq 2\mathsf{OPT}$

Proof. Let F^* an optimal solution. Notice that the graph $(V, E \setminus F^*)$ can be thought of as k components $V_1, \ldots, V_k$ where V_i is a connected component containing t_i. For each i, let F_i^* be the edges with exactly one endpoint in V_i, so removing F_i^* disconnects t_i from $T \setminus t_i$.

From the fact that F_i is a minimum cut, we must have $w(F_i) \leq w(F_i^*)$, so $\sum_i w(F_i) \leq \sum_i w(F_i^*)$. Now consider the cost of the term on the RHS. Each edge in F^* only appears in exactly two terms in the sum, so we have that $\sum_i w(F_i^*) \leq 2 \sum_{e \in F^*} w(e) = 2\mathsf{OPT}$.

$\square$