CMPUT 675: Approximation Algorithms **Fall 2013**

Lecture 8, 9 (Oct 1 and 3): k-median, and Steiner forest

Lecturer: Mohammad R. Salavatipour *Scribe: based on older notes*

8.1 k-median problem

k-median is an important clustering problem that has similarities to both k-center and facility location problem. An instance of this problem is similar to k-center: a complete graph $G = (V, E)$ with metric edge costs $c(e)$ and a positive integer k. The difference is in the objective function: now instead of minimazing maximal distance from nodes to the nearest center ($\max\limits_{v \in V} d(v, S)$) we're minimizing the sum of distances from nodes to the nearest open center ($\min \sum\limits_{v \in V} d(v, S)$).

k-median problem

- Input
  - A complete graph $G = (V, E)$ with set of clients $D \subseteq V$ and facilities $F \subseteq V$
  - c_{ij} is the cost of assigning location j to a facility at location i (and this cost function is metric)
  - k - the maximal number of facilities we can open.
- Goal: find $F' \subseteq F$, where $|F'| \leq k$, to open, s.t. the assignment cost is minimized: $\min \sum\limits_{j \in N} d(j, F') = kmed(F')$.

Without loss of generality, we can assume that $|F'| = k$. As in the k-center problem, we assume that the distance matrix is symmetric, satisfies triangle inequality, and has zeros on the diagonal. We present a local search algorithm for k-median problem with good approximation ratio. For every subset $F' \subseteq F$ we use $kmed(F')$ to denote the cost of the solution if set F' is chosen.

Local search algorithm

1. Start from an arbitrary F' with $|F'| = k$
2. On each iteration see see if swapping a facility in F' with one in $F - F'$ improves the solution
3. Iterate until no single swap yields a better solution

Figure 8.1: Local search algorithm for k-median problem

Theorem 1 *If F' is a local optimum and F^* is a global optimum, then* $\text{kmed}(F') \leq 5\text{kmed}(F^*)$

Proof. Our proof is based on "Simpler Analyses of Local Search Algorithms for Facility Location" by Gupta and Tangwongsan (arXiv:0809.255). The proof will focus on constructing a set of special swaps. These swaps will all be constructed by swapping into the solution location i^* in F^* and swapping out of the solution one location i' in F'. Each $i^* \in S^*$ will participate in exactly one of these k swaps, and each $i' \in F'$ will participate in at most 2 of these k swaps. We will allow the possibility that $i^* = i'$, and hence the swap move is degenerate, but clearly such a "change" would also not improve the objective function of the current solution, even if we change the corresponding assignment. For any two points i, j we use $d(i, j)$ to refer to the distance between these two points, i.e. c_{ij}. Let $\phi : F^* \to F$ be a mapping that maps each $f^* \in F^*$ to the nearest facility in F, i.e. $d(f^*, \phi(f^*)) \leq d(f^*, f)$ for all $f \in F'$.

Let $R \subseteq F'$ be those that have at most one $f^* \in F^*$ mapped to them. Now we define a set of k pairs of potential swaps: $S = \{(v, f^*) \subseteq R \times F^*\}$ such that:

1. $\forall f^* \in F^*$, it appears in exactly one pair $(v, f^*) \in S$.
2. each node $r \in R$ with $\phi^{-1}(r) =$ appears in at most two swaps.
3. each nodr $r \in R$ with $\phi^{-1}(r) = f^*$ appears only in one swap.

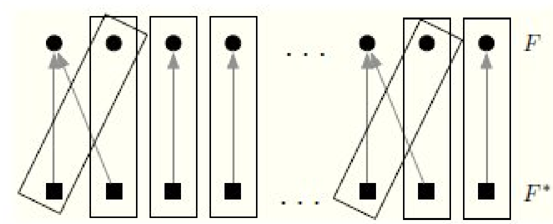

Figure 8.2: An example of mapping $\phi : F^* \to F$

How to build this set S? for each $r \in R$ with in-degree 1 we add pairs $(r, \phi^{-1}(r))$ to S. Let F_1^* be those of F^* that are matched this way. Other facilities in R have in-degree zero; let us call this set R_0. Note that

$$|F^* \setminus F_1| \leq 2|R_0|.$$

Now we can add other pairs by arbitrarily matching each node of R_0 with at most two in $F^* \setminus F_1^*$.

Observation: For any pair $(r, f^*) \in S$ and $\tilde{f}^* \in F^*$ with $\tilde{f}^* \neq f$: $\phi(\tilde{f}^*) \neq r$.

We use the fact that none of these potential swaps (in S) are improving to derive a bound on the cost of local optimum. Suppose that $\sigma : D \to F'$ and $\sigma^* : D \to F^*$ are mappings of clients to facilities in the local optimum and global optimum, respectively. For each $j \in D$, let $O_j = d(j, F^*) = d(j, \sigma^*(j))$ be the cost of connecting j in the optimum solution and $A_j = d(j, F') = d(j, \sigma(j))$ be its cost in the local optimum. We use $N^*(f^*) = \{j | \sigma^*(j) = f^*, f^* \in F^*\}$ to denote those assigned to f^* in the optimum solution and $N(f) = \{j | \sigma(j) = f, f \in F'\}$ to denote those assigned to f in the local optimum.

Lemma 1 *For each swap* $(r, f^*) \in S$*:*

$$\text{kmed}(F' + f^* - r) - \text{kmed}(F') \leq \sum_{j \in N^*(f^*)} (O_j - A_j) + \sum_{j \in N(r)} 2O_j.$$

Proof. Suppose we do the swap (r, f^*) and let's see how much the cost increases (note that since we are at a local optimum, this must be the case). We can upper bound this by giving a specific assignment of clients to facilities. Clearly the optimum assignment of clients to facilities cannot cost more than this:

- each client of $N^*(f^*)$ is assigned to f^*
- each client $j \in N(r) \setminus N^*(f^*)$ is assigned by the following rule: suppose $\tilde{f}^* = \sigma(j)$; we assign j to $\tilde{f} = \phi(\tilde{f}^*)$. Note that $\tilde{f} \neq r$.
- the assignment of all other clients remain unchanged.

For each $j \in N^*(f^*)$ the change in cost is exactly $O_j - A_j$; summing this over all $j \in N^*(f^*)$ gives the first term on the RHS. For $j \in N(r) \setminus N^*(f^*)$, the change in cost is:

$$
\begin{array}{rcll}
d(j,\tilde{f}) - d(j,r) & \leq & d(j,\tilde{f}^*) + d(\tilde{f}^*,\tilde{f}) - d(j,r) & \text{using triangle inequality} \\
& \leq & d(j,\tilde{f}^*) + d(\tilde{f}^*,r) - d(j,r) & \text{since } \tilde{f} \text{ is closest to } \tilde{f}^* \\
& \leq & d(j,\tilde{f}^*) + d(j,\tilde{f}^*) & \text{using triangle inequality} \\
& = & 2O_j &
\end{array}
$$

Thus, summing up the total change for all these clients is at most: $\sum_{j\in N(r)\setminus N^*(f^*)} 2O_j \leq \sum_{j\in N(r)} 2O_j$. ∎

Now we use this lemma and sum over all pairs $(r, f^*) \in S$. Note that each $f^* \in F^*$ appears exactly once and each $r \in R \subseteq F'$ appears at most twice. Therefore:

$$
\begin{array}{rcl}
\displaystyle\sum_{(r,f^*)\in S} (\text{kmed}(F' + f^* - r) - \text{kmed}(F')) & \leq & \displaystyle\sum_{f^*\in F^*} \sum_{j\in N^*(f^*)} (O_j - A_j) + 2\sum_{r\in R}\sum_{j\in N(r)} 2O_j \\
& \leq & \text{cost}(F^*) - \text{cost}(F') + 4\text{cost}(F^*)
\end{array}
$$

This implies that $\text{cost}(F') \leq 5\text{cost}(F^*)$.

∎

Note that the running time of this algorithm is not necessarily polynomial. To get polynomial time algorithm we only consider swaps which improve the cost by a factor of at least $(1 + \delta)$ for some $\delta > 0$. So when the algorithm stops we are in an almost locally optimum solution, i.e. each potential swap can only improve by a factor of smaller than $1 + \delta$. Then essentially the same analysis shows that the approximation ratio of the algorithm is at most $5(1 + \delta)$ which is $5 + \epsilon$ for sufficiently small $\epsilon > 0$. If the objective value is M for the optimum solution then the algorithm takes at most $O(\log_{1+\delta} M)$ steps to arrive at a locally optimum solution which is polynomial.

Improvment using t-swaps: A similar anaylsis shows that if one considered all t-swaps (instead of just 1-swaps) for a constant value of t at each step then the local search has a ratio of $3 + \frac{2}{t}$.

8.2 Generalized Steiner tree problem (Steiner forest)

We now turn to the problem known as generalized Steiner tree problem

Generalized Steiner tree problem

- Input
  - An undirected graph $G = (V, E)$.
  - Cost $c : E \to Q^+$.
  - Collection of disjoint sets $S = \{S_1, ..., S_k\}$, $S_i \subseteq V$.

- Goal: find a minimum cost subgraph of G, s.t. for every i each pair of vertices of S_i is connected.

Let us restate the problem; this will also help generalize it later. Define a connectivity requirement function r that maps unordered pairs of vertices to $\{0, 1\}$ as follows:

$$r(u,v) = \begin{cases} 1, & \text{if } u, v \text{ belong to the same set } S_i \\ 0, & \text{otherwise} \end{cases}$$

Now, the problem is to find a minimum cost subgraph that contains a u-v path for each pair (u, v) with $r(u, v) = 1$. In general, the solution will be a forest.

Every pair $(S, \bar(S))$ defines a cut. Let $\delta(S)$ be edges of this cut. Let us define a function on all cuts in G, $f : 2^V \to \{0, 1\}$, which specifies the minimum number of edges that must cross each cut in any feasible solution.

$$\forall S : f(S) = \begin{cases} 1, & \text{if } \exists u \in S, v \in \overline{S} \text{ where } r(u,v) = 1 \\ 0, & \text{otherwise} \end{cases}$$

Let us also introduce a 0/1 variable x_e for each edge $e \in E$;x_e will be set to 1 iff e is picked in the subgraph.

The integer LP program is:

$$\begin{array}{rccc} \textbf{Integer LP} \quad \min & \sum\limits_{e \in E} c(e) \cdot x_e & & \\ \text{s.t. } \forall S \subseteq V : & \sum\limits_{e \in \delta(S)} x_e & \geq & f(s) \\ & x_e \in \{0, 1\} & & \end{array}$$

By relaxing the integral constraints we obtain the following Primal and the corresponding dual LP:

- **Primal:**

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} C_e x_e, \\ \text{such that} & \forall S \subseteq V, \sum_{e \in \delta(S)} x_e \geq f(S) \\ & x_e \geq 0. \end{array}$$

- **Dual:**

$$\begin{array}{ll} \text{maximize} & \sum_{S \subseteq V} f(S) y_s, \\ \text{such that} & \forall e \in E, \sum_{S : e \in \delta(S)} y_S \leq C_e \\ & y_S \geq 0. \end{array}$$

Say edge e is tight if $\sum\limits_{S : e \in \delta(S)} y_s = C_e$. We use a primal-dual schema for our approximation algorithm. Like other primal-dual algorithms, in each iteration, we improve the dual solution and make primal solution more feasible. We start with an infeasible (all zero) primal solution and feasible zero dual solution. We are improving the dual solution by increasing y_S for some S with $f(S) = 1$ (because if $f(S) = 0$, we could not improve the objective function) until a constraint becomes tight. Then, we choose the edges corresponding to the tight constraints in primal solution and update the dual solution and progress until reaching a feasible primal solution is found. Let us state the primal and relaxed dual complementary slackness conditions. The algorithm will pick edges integrally only. Define the degree of set S to be the number of picked edges crossing cut $(S, \overline{S})$

Primal: For each $e \in E, x_e \neq 0 \Rightarrow \sum\limits_{S : e \in \delta(S)} y_s = c_e$. Equivalently, every picked edge must be tight

Relaxed dual conditions: The following relaxation of the dual conditions would have led to a factor 2 algorithm: for each $S \subseteq V, y_s \neq 0 \Rightarrow \sum_{e: e \in \delta(s)} x_e \leq 2f(S)$, i.e., every raised cut has degree at most 2. However, we do not know how to ensure this condition. But we can still obtain a 2-approximation algorithm by relaxing this conidition further.

Definition 1 *Given an assignment of x_e and y_s values:*

- *A set $S \subseteq V$ is unsatisfied if $f(S) = 1$ and no edges from $\delta(S)$ are picked.*
- *A set $S \subseteq V$ is active if it's a minimal (inclusion-wise) unsatisfied set.*

Lemma 2 *Set S is active iff it is a connected component in the currently picked forest and $f(S) = 1$.*

Proof. Consider an active set S. Suppose that S is a part of a connected component. Then there is at least one edge in $\delta(S)$ that is picked. Hence, S would be satisfied which is a contradiction. Thus, S contains more than 1 connected component. Since $f(S) = 1$ there's at least one vertex v in S that needs to be connected to outside of S. Suppose v belongs to a component C_i. Then $f(C_i) = 1$ and C_i is unsatisfied which violates minimality of S. ■

The algorithm for Steiner tree problem is presented below:

Algorithm for generalized Steiner tree problem

1. $F \leftarrow \emptyset$
2. $y_s \leftarrow \emptyset$ for all S
3. while there's unsatisfied set do
4. simultaneously raise y_s for each active set S, until some edge e goes tight.
5. $F \leftarrow F \cup \{e\}$
6. return $F' = \{e \in F | F - \{e\}$ is primal infeasible$\}$

Figure 8.3: LP based algorithm for generalized Steiner tree problem

Figure 8.1 shows a sample run of the algorithm. Suppose we have two disjoint subsets $S_1 = \{u, v\}$ and $S_2 = \{s, t\}$. At the beginning of the algorithm, $\{u\}, \{v\}, \{s\}, \{t\}$ are four active sets, each of which contains one vertex only. The algorithm raises their y_S values simultaneously, and stops at the value of 6 when edge ua and bv become tight. One of them, say ua is picked, and the iteration ends. In the second iteration, $\{u, a\}$ replaces $\{u\}$ as an active set and the algorithm finds already tight edge bv. Then algorithm updates active sets and raise value of their variables, and continues. Figure below shows the final result of running the algorithm. In this figure, active sets are shown with a closed area containing them and the final value of their variables are depicted beside the boundary of these closed areas. The bold edges are the edges added to F in the loop. At the end, all edges in F except the redundant edge ua are added to F' and returned.

Lemma 3 *The Primal-Dual Steiner forest algorithm (given above) has a polynomial running time.*

Proof. First, note that the algorithm can choose at most $|V| - 1$ edges and terminates after at most $|V| - 1$ iterations. It is trivial that there exists an unsatisfied set if and only if there exists a minimal unsatisfied set (*i.e.* an active set). Therefore, by finding active sets, we could also check the condition of loop. To find active sets, by Lemma 2, it is enough to find connected components of $G' = (V, F)$ and check their f value. Thus, we could implement the loop in polynomial time. Finally, to find F', we eliminate all edges which are not in

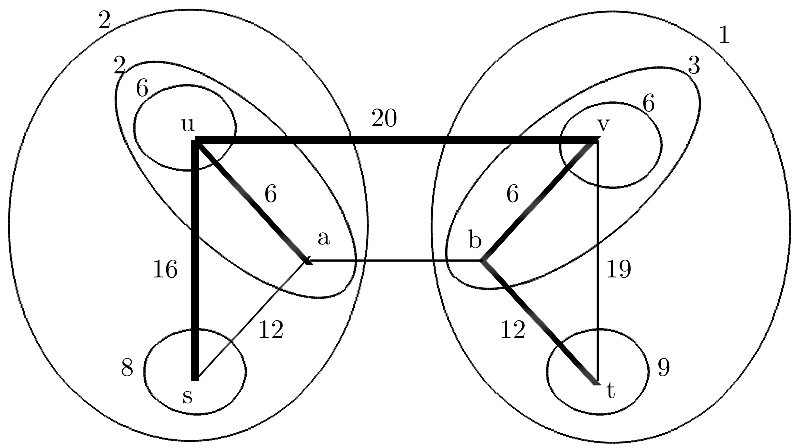

Figure 8.1: A sample run of algorithm STEINER-FOREST$(G, \mathcal{S})$.

any path between two vertices from some S_i, which could be implemented in polynomial time, too (because $G' = (V, F)$ is a forest and there is at most one path between two vertices). ■

Since the loop terminates when there is not any unsatisfied set, F is a feasible primal solution. In addition, third step of the algorithm keeps feasibility. Therefore, this algorithms yields a feasible solution for Steiner forest problem in polynomial time.

Lemma 4 *At the end of the algorithm, the pruned edge set F' and dual variables y are feasible primal and dual solutions, respectively.*

Proof. The algorithm continues until all the sets are satisfied. Therefore set F (before pruning) is clearly a feasible primal solution. In each iteration, dual variables of connected components only are raised. Therefore, no edge running within the same component can go tight, and so F is acyclyc, i.e., it is a forest. Also, by definition F' is feasible too. So we have a feasible primal solution. Now we show that the dual is also a feasible solution. In Step 2 of the algorithm, only edges between two components are raised and whenever a constraint becomes tight it will be de-activated (and edge between that component and another is added to the solution; so it no longer becomes an active set). Therefore no dual constraint will be violated. ■

Let $deg_{F'}(s)$ denote the number of edges of F' crossing the cut $(S, \overline{S})$.

Lemma 5 *Let C be any connected component (w.r.t currently selected edges) in any iteration of algorithm. If $f(C) = 0$, then $deg_{F'}(C) \neq 1$ (*i.e.* it is either 0 or ≥ 2), where $deg_{F'}$ denotes the number of edges from F' that cross the cut (S, S').*

Proof. If $deg_{F'}(C) = 1$, there exists a unique edge e crossing the cut (S, S'). Consider the pair of vertices u and v that are connected through a path containing e and $r(u, v) = 1$. Such a pair should exist otherwise edge e can be safely removed, which contradicts with non-redundancy of edges in F'. Therefore, e connects one vertex in C to another vertex in C'. This reaches us to contradiction because having $r(u, v) = 1$ means that $f(C) = 1$. ■

Lemma 6 $\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S$

Proof. The algorithm picks an edge when it becomes tight. Considering the primal slackness condition:

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \left(\sum_{S: e \in \delta(S)} y_S \right).$$

$$\sum_{e \in F'} c_e = \sum_{S \subseteq V} \left(\sum_{e \in \delta(S) \cap F'} y_S \right) = \sum_{S \subseteq V} deg_{F'}(S).y_S.$$

Therefore, we need to prove the following inequality:

$$\sum_{S \subseteq V} deg_{F'}(S).y_S \leq 2 \sum_{S \subseteq V} y_S. \tag{8.1}$$

We will show that in each iteration of the algorithm the amount of increase in the right-hand side of the inequality 8.1 is more than the corresponding increase in the left-hand side. Consider an arbitrary iteration of the algorithm and let $\triangle$ be the amount of increase in the y_S variables. Since in each iteration only the dual variables of active sets are increased, we need to show:

$$\triangle \times \left(\sum_{\text{active } S} deg_{F'}(S) \right) \leq 2 \triangle \cdot (\#\text{of active sets}).$$

Consider the graph, called H, on the same vertex set and edge set as F'. For every set of vertices of a connected component with respect to F, contract all those vertices in H into one single (big) vertex. Delete isolated vertices. Call this new graph H'. Note that H' is a forest and that the degree of each (big) vertex in H' is the same as the degree of corresponding set of vertices that are contracted. Each vertex of H' that corresponds to an active set has non-zero degree. By Lemma **??**, the degree of every vertex in H' that corresponds to a non-active set is at least 2. Also, because H' is a forest, its average degree is at most 2. Therefore, the average degree of nodes of H' that correspond to active sets is at most 2. ■

It follows from the above lemmas that:

Theorem 2 *The primal-dual algorithm is a 2-approximation for Steiner forest.*

Tight Example: A cycle with n nodes and connectivity requirement of 1 has an integrality gap of $2 - \frac{2}{n}$; an optimal fractional solution pickes each edge with fractional value $\frac{1}{2}$ for a total of $n/2$ whereas any integer solution is a path with total cost $n - 1$.