Lecture 2: Local Search

1 Max-Cut

Notation: (A, B) is used to denote the set of edges with one endpoint in A and the other in B.

Procedure LocalSearch-MC

1: $(S, \overline{S})$ will be the cut returned. Initialize S to any arbitrary set of vertices.
   Initialize `Loc-Opt` to `false`.
2: **while** `Loc-Opt` is `false` **do**
3: Set `Loc-Opt` to `true`.
4: If there exists vertex $v \in S$ (or $v \in \overline{S}$) such that $w(S - v, \overline{S} + v) > w(S, \overline{S})$ (respectively, $w(S + v, \overline{S} - v) > w(S, \overline{S})$), assign $S = S - v$ (respectively, $S = S + v$).
   `Loc-Opt` is set to `false`.
5: **end while**

The procedure above terminates since the weight of the cut increases in each iteration. Let $(S, \overline{S})$ be the locally-optimal cut returned by the algorithm. $\texttt{alg} = \sum_{v \in S} w(v, \overline{S}) = \sum_{v \in \overline{S}} w(v, S)$. We know

$$\forall v \in S : \quad w(v, S) \leq w(v, \overline{S}); \qquad\qquad \forall v \in \overline{S} : \quad w(v, \overline{S}) \leq w(v, S)$$

Thus, $4\texttt{alg} \geq \sum_{v \in S}(w(v, S) + w(v, \overline{S})) + \sum_{v \in \overline{S}}(w(v, S) + w(v, \overline{S})) = 2w(E)$, giving us $\texttt{alg} \geq w(E)/2 \geq \texttt{opt}/2$.

Directed Graphs. In directed graphs, we use the notation (A, B) to denote the arcs with tail in A and head in B. The local search algorithm for finding the maximum cut in a digraph is similar, except in the end we return the cut $(S, \overline{S})$ or $(\overline{S}, S)$, whichever is better.

Suppose the algorithm terminates with the partition $(S, \overline{S})$. Note that $\texttt{alg} \geq w(S, \overline{S}) = \sum_{v \in S} w(v, \overline{S}) = \sum_{v \in \overline{S}} w(S, v)$. Local optimality gives

$$\forall v \in S : \quad w(S \setminus v, v) \leq w(v, \overline{S}); \qquad\qquad \forall v \in \overline{S} : \quad w(v, \overline{S} \setminus v) \leq w(S, v)$$

In the homework, you are asked to prove that the weight of the cut returned by the algorithm is at least $w(E)/4$; thus it is a 1/4-approximation. You are also asked to come up with an example for which the weight of the cut returned is exactly $w(E)/4$. Does this imply we can't analyze the algorithm better? No. This is because maybe `opt` is much less than m in these cases. Let us analyze the performance of our algorithm with respect to `opt`.

Let $(O, \overline{O})$ be the optimal dicut. We can write `opt` as follows

$$\mathtt{opt} = \sum_{v \in \overline{O}} w(O,v) = \sum_{v \in \overline{O} \cap S} w(O \cap S, v) + \sum_{v \in \overline{O} \cap S} w(O \cap \overline{S}, v) + \sum_{v \in \overline{O} \cap \overline{S}} w(O \cap S, v) + \sum_{v \in \overline{O} \cap \overline{S}} w(O \cap \overline{S}, v) \tag{1}$$

It'll be useful to draw the rectangle divided in four quadrants picture - I am too lazy to do it in the notes. Now we use the local optimality conditions above for the first and the fourth term in (1) to give

$$\sum_{v \in \overline{O} \cap S} w(O \cap S, v) \leq \sum_{v \in \overline{O} \cap S} w(v, \overline{S}) = w(\overline{O} \cap S, \overline{S}) \leq w(S, \overline{S}) \tag{2}$$

$$\sum_{v \in \overline{O} \cap \overline{S}} w(O \cap \overline{S}, v) = \sum_{v \in O \cap \overline{S}} w(v, \overline{O} \cap \overline{S}) \leq \sum_{v \in O \cap \overline{S}} w(S, v) = w(S, O \cap \overline{S}) \tag{3}$$

The second and third term in (1) are bounded as

$$\sum_{v \in \overline{O} \cap S} w(O \cap \overline{S}, v) \leq \sum_{v \in S} w(\overline{S}, v) = w(\overline{S}, S) \tag{4}$$

$$\sum_{v \in \overline{O} \cap \overline{S}} w(O \cap S, v) \leq \sum_{v \in \overline{O} \cap \overline{S}} w(S, v) = w(S, \overline{O} \cap \overline{S}) \tag{5}$$

Putting all together, gives

$$\mathtt{opt} \leq w(S, \overline{S}) + w(\overline{S}, S) + w(S, \overline{O} \cap \overline{S}) + w(S, O \cap \overline{S}) = 2w(S, \overline{S}) + w(\overline{S}, S) \leq 3\mathtt{alg}.$$

2 Metric Facility Location and k-Median

Procedure LocalSearch-UFL

1: X is the set of facilities opened initialized to an arbitrary facility. Clients always connect to the closest facility in X. Initialize `Loc-Opt` to `false`.
2: **while** `Loc-Opt` is `false` **do**
3: Set `Loc-Opt` to `true`.
4: *(Add):* If adding a facility $i \in F \setminus X$ decreases the total cost, $X = X \cup i$, and `Loc-Opt` is set to `false`.
5: *(Delete):* If deleting a facility $i \in X$ decreases the total cost, $X = X - i$, and `Loc-Opt` is set to `false`.
6: *(Swap):* If swapping a facility $i \in X$ with $i' \in F \setminus X$, decreases the total cost, $X = X - i + i'$, and `Loc-Opt` is set to `false`.
7: **end while**

Let X be the set of facilities opened at the end of the above algorithm. We use notation as in the previous class. $\sigma(j)$ will denote the facility in X client j is connected to. $\Gamma(i)$ denotes the set of clients connected to facility i. X^* will denote the set of facilities opened in the optimal solution. σ^* and Γ^* are defined respectively. $c_j = c(\sigma(j), j)$ and $c^*_j = c(\sigma^*(j), j)$. $F_{\mathtt{alg}} = \sum_{i \in X} f_i$, $C_{\mathtt{alg}} = \sum_{j \in C} c_j$. Similarly F^* and C^* are defined.

Bounding $C_{\texttt{alg}}$. We know that adding any facility $i \in X^* \setminus X$ doesn't decrease cost. Note that if we add such an i, we could've moved all clients in $\Gamma^*(i)$ to i. Since this doesn't decrease cost,

$$\forall i \in X^* \setminus X; \qquad \sum_{j \in \Gamma^*(i)} c_j \le f_i + \sum_{j \in \Gamma^*(i)} c_j^*$$

Note that the above is true for $i \in X^* \cap X$ since j goes to the closest facility in X. So, adding over all $i \in X^*$ we get, $\sum_{i \in X^*} \sum_{j \in \Gamma^*(i)} c_j \le \sum_{i \in X^*} f_i + \sum_{j \in \Gamma^*(i)} c_j^*$, that is,

$$C_{\texttt{alg}} \le F^* + C^*$$

Bounding $F_{\texttt{alg}}$. Fix an $i \in X$. How much can the connection cost of clients increase if i is deleted? All clients in $\Gamma(i)$ will move to their second-nearest facility in X. But what handle do we have on the distance between j and the second-nearest facility?

Well we know the cost to connect j and $\sigma^*(j)$. So, if $\sigma^*(j)$ is in X, let's move j to that. What if $\sigma^*(j)$ isn't there? Well move it to the facility in X closest to $\sigma^*(j)$. This motivates the following definition.

Given $i^* \in X^*$, let $\texttt{nearest}(i^*)$ denote the facility i in X with minimum $c(i, i^*)$. Let's get back to our facility i. Let us look at clients $j \in \Gamma(i)$ such that $\texttt{nearest}(\sigma^*(j)) \ne i$. Call these clients $\texttt{Far}(i)$. These clients we can argue about. If i is deleted, then clients in $\texttt{Far}(i)$ can be assigned to $\texttt{nearest}(\sigma^*(j))$, and their new connection cost becomes

$$\begin{aligned} c(\texttt{nearest}(\sigma^*(j)), j) \quad &\le \quad c(\sigma^*(j), j) + c(\sigma^*(j), \texttt{nearest}(\sigma^*(j))) \\ &\le \quad c_j^* + c(\sigma^*(j), \sigma(j)) \quad \le \quad c_j^* + c(\sigma^*(j), j) + c(j, \sigma(j)) = c_j + 2c_j^* \end{aligned}$$

The second inequality follows from the definition of $\texttt{nearest}()$, and all others from metricity. So, for all these clients, the connection cost increases by at most $2c_j^*$.

What about the other clients in $\Gamma(i)$? These clients j have $\texttt{nearest}(\sigma^*(j)) = i$. In fact, let all these $\sigma^*(j)$'s be called the set X_i^*. Formally,

$$X_i^* := \{i^* \in X^* : i^* = \sigma^*(j) \text{ for some } j \in \Gamma(i), \ \text{ and } \texttt{nearest}(i^*) = i\}. \tag{6}$$

A simple but crucial observation is that X_i^*'s are disjoint for different $i \in X$ (since $\texttt{nearest}(i^*) = i$ for $i^* \in X_i^*$). Let $\texttt{friend}(i)$ be the facility in X_i^* closest to i and consider swapping i and $\texttt{friend}(i)$. All clients in $\texttt{Far}(i)$ go and connect to $\texttt{nearest}(\sigma^*(j))$. All clients in $\Gamma(i) \setminus \texttt{Far}(i)$ connect to $\texttt{friend}(i)$. Since this swap can't increase cost, we have

$$f_i + \sum_{j \in \Gamma(i)} c_j \le f_{\texttt{friend}(i)} + \sum_{j \in \texttt{Far}(i)} \left(c_j + 2c_j^*\right) + \sum_{j \in \Gamma(i) \setminus \texttt{Far}(i)} c(\texttt{friend}(i), j) \tag{7}$$

Now, $c(\texttt{friend}(i), j) \le c(i,j) + c(i, \texttt{friend}(i)) \le c(i,j) + c(i, \sigma^*(j)) \le c_j + c(i,j) + c(\sigma^*(j), j) = 2c_j + c_j^*$. Thus,

$$f_i + \sum_{j \in \Gamma(i)} c_j \le f_{\texttt{friend}(i)} + \sum_{j \in \texttt{Far}(i)} \left(c_j + 2c_j^*\right) + \sum_{j \in \Gamma(i) \setminus \texttt{Far}(i)} \left(2c_j + c_j^*\right)$$

This implies, $f_i \leq f_{\texttt{friend}(i)} + \sum_{j\in\Gamma(i)}(2c_j^* + c_j)$ (weak!). Adding over all $i \in X$, and using the fact that X_i^*'s are disjoint, gives us

$$F_{\texttt{alg}} = \sum_{i\in X} f_i \leq \sum_{\texttt{friend}(i)\in X_i^*} f_{i^*} + 2C^* + C_{\texttt{alg}} \leq 2F^* + 3C^*$$

where we use the bound on $C_{\texttt{alg}}$.

$$\texttt{alg} = F_{\texttt{alg}} + C_{\texttt{alg}} \leq 3F^* + 4C^*$$

The above local search algorithm is a 4-approximation.

k-Median

(Some more details of what I was talking about in the last 25 minutes in class)

Procedure LocalSearch-kMed

1: X is the set of facilities opened initialized to any set of k arbitrary facility. Clients always connect to the closest facility in X. Initialize `Loc-Opt` to `false`.
2: **while** `Loc-Opt` is `false` **do**
3: Set `Loc-Opt` to `true`.
4: *(Swap):* If swapping a facility $i \in X$ with $i' \in F \setminus X$, decreases the total cost, $X = X - i + i'$, and `Loc-Opt` is set to `false`.
5: **end while**

We use notation as in the case of UFL. Note that there is no $F_{\texttt{alg}}$ and F^*. However, we cannot bound $C_{\texttt{alg}}$ as we did in the UFL case since we cannot add facilities.

Consider a facility $i \in X$, and define X_i^* and $\texttt{friend}(i) \in X_i^*$ as in (6). From the previous analysis, we can guess that we would want to analyze the case of swapping i with $\texttt{friend}(i)$. However, a little thought shows that this doesn't work if $|X_i^*| > 1$. In particular, we cannot argue about the connection costs for j with $\texttt{nearest}(\sigma^*(j)) = i$ but $\sigma^*(j) \neq \texttt{friend}(i)$. The trick is *not* to "swap-out" such facilities at all.

More definitions. Let $X_0 := \{i \in X : |X_i^*| = 0\}$, $X_1 := \{i \in X : |X_i^*| = 1\}$, and $X_2 := \{i \in X : |X_i^*| \geq 2\}$. (Note that if there are no facilities in X_2, we are in the *lucky world*, as we talked about in class). Since $|X| = |X^*| = k$, and since X_i^*'s are disjoint for different $i \in X$, we have that $|X_0| \geq |X_2|$. This lets us define the following swap pairs. We will perform a swap of the facilities in the swap pairs, and since they cannot help, we will be bounding connection costs. We use the following definition: given a set $R \subseteq X^* \times X$, the degree of $i^* \in X^*$ is $deg(i^*) := |\{i : (i^*, i) \in R\}|$. Similarly degree of $i \in X$ is defined.

Claim 1. *There exists a set $R \subseteq X^* \times (X_0 \cup X_1)$ such that for all $i^* \in X^*$, $deg(i^*) = 1$, for all $i \in X_1$, $deg(i) = 1$, for all $i \in X_0$, $deg(i) \leq 2$.*

Proof. For all $i \in X_1$, add $(\texttt{friend}(i), i)$ to R. Now arbitrarily map the remaining $k - |X_1|$ facilities of X^* with facilities in X_0. Since $k - |X_1| = |X_0| + |X_2| \leq 2|X_0|$, we can always find one which maps $i \in X$ with at most 2 facilities in X^*. □

The above claim says that each $i^* \in X^*$ is swapped in once, and each facility in X_0 are swapped out at most twice.

Now let us look at the swaps defined by R: for $(i^*, i) \in R$, add i^* in and delete i. For each $j \in \Gamma^*(i^*)$, we re-assign it to i^*. If $i \in X_0$, we assign each $j \in \Gamma(i) \setminus \Gamma^*(i^*)$ to $\texttt{nearest}(\sigma^*(j))$. If $i \in X_1$, we assign each $j \in \texttt{Far}(i)$ to $\texttt{nearest}(\sigma^*(j))$, and we know for each $j \in \Gamma(i) \setminus \texttt{Far}(i)$, we have $\sigma^*(j) = \texttt{friend}(i) = i^*$. That is, $\texttt{Far}(i) = \Gamma(i) \setminus \Gamma^*(i^*)$. Since swaps don't decrease cost, we get the following:

If $i \in X_0 \cup X_1$

$$\sum_{j \in \Gamma^*(i^*)} c_j + \sum_{j \in \Gamma(i) \setminus \Gamma^*(i^*)} c_j \quad \leq \quad \sum_{j \in \Gamma^*(i^*)} c_j^* + \sum_{j \in \Gamma(i) \setminus \Gamma^*(i^*)} (2c_j^* + c_j)$$

implying,

$$\sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) \leq \sum_{j \in \Gamma(i)} 2c_j^*.$$

Thus,

$$\sum_{(i^*,i) \in R} \sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) \leq \sum_{(i^*,i) \in R} \sum_{j \in \Gamma(i)} 2c_j^*$$

The LHS is precisely $\sum_{i^* \in X^*} deg(i^*) \cdot \left(\sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) \right) = C_{\texttt{alg}} - C^*$. The RHS is at precisely $\sum_{i \in X_0 \cup X_1} deg(i) \cdot \left(\sum_{j \in \Gamma(i)} 2c_j^* \right)$. which is at most $4C^*$. This implies a 5-approximation.

p swaps: the state of the art for k-Median.

How can we do better than 5? Well, this brings us to the idea of swapping more than one facility at a time (this was brought up in the class). Suppose I can swap p facilities in-and-out at a time (p is a constant). Well, instead of defining swap pairs, we will have swap sets. That is, each entry of R would be (A^*, A) with $|A^*| = |A| \leq p$. We still define $deg(i^*) = |\{(A^*, A) \in R : i^* \in A^*\}|$, and similarly, $deg(i)$.

Let $X_t := \{i \in X : |X_i^*| = t\}$. Note that $|X_0| = \sum_{t \geq 1} (t-1)|X_t|$ since $\sum_{t \geq 0} |X_t| = \sum_{t \geq 0} t|X_t| = k$. For each $t \geq 1$, and for each $i \in X_t$, assgin $(t-1)$ arbitrary distinct facilities of X_0 to i. The above equality tells us that this can be done. We describe the swap sets now.

For $1 \leq t \leq p$, and for $i \in X_t$, form the swap set $A_i^* = X_i^*$ and A_i being i and the $(t-1)$ facilities of X_0 assigned to it. Add p copies of (A_i^*, A_i) to R.

For $t > p$, for $i \in X_t$ we the sets A_i will *not* contain i. Thus, facilities in X_{p+1} or larger are never swapped out. Instead, consider the t facilities in X_i^* and pick t subsets of size exactly p such that each $i^* \in X_i^*$ appears in exactly p subsets. This can be done in many ways. Similarly, consider the $(t-1)$ facilities of X_0 assigned to i and for t subsets of size exactly p such that each facility appears in at most $(p+1)$. Pair these up arbitrarily and add it to R.

This completes the description of R. It should be clear that $deg(i^*) = p$ for all $i^* \in X^*$, $deg(i) \leq p+1$ for all $i \in X_0 \cup \cdots \cup X_p$ and $deg(i) = 0$ for al $i \in X_{p+1} \cup \cdots$. The following should also be clear from the analysis of the 5 factor and the sets that we add.

Claim 2. *For any* $(A_i^*, A_i) \in R$,

$$\sum_{i^* \in A_i^*} \sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) \leq \sum_{i' \in A_i} \sum_{j \in \Gamma(i')} 2c_j^*$$

Proof. Consider (A_i^*, A_i) for $i \in X_1 \cup \cdots \cup X_p$ first. Firstly, for all $i^* \in A_i^*$, assign clients in $\Gamma^*(i^*)$ to i^*. Consider a facility i' deleted in A_i. Let $\Gamma'(i')$ be the set of clients in $\Gamma(i)$ which haven't been assigned to a client in $\Gamma^*(i^*)$ for some $i^* \in A_i^*$. Note that by our choice of A_i, all these clients j can be assigned to $\texttt{nearest}(\sigma^*(j))$. Thus, we get the following inequality

$$\sum_{i^* \in A_i^*} \sum_{j \in \Gamma^*(i^*)} c_j \;+\; \sum_{i' \in A_i} \quad \sum_{j \in \Gamma'(i')} c_j \;\leq\; \sum_{i^* \in A_i^*} \sum_{j \in \Gamma^*(i^*)} c_j^* \;+\; \sum_{i' \in A_i} \quad \sum_{j \in \Gamma'(i')} (c_j + 2c_j^*)$$

Rearranging, we get the claim. For (A_i^*, A_i) for $i \in X_t$, $t > p$, note that we swap out facilities only in X_0. So the above inequality holds for those as well. □

Theorem 1. *The* p*-swap local search algorithm for* k*-median is a* $(3 + \frac{2}{p})$*-approximation.*

Proof. Add the inequalities given in the above claim for all swap sets in R. We get in the LHS

$$\sum_{(A_i^*, A_i) \in R} \sum_{i^* \in A_i^*} \sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) = \sum_{i^* \in X^*} deg(i^*) \left(\sum_{j \in \Gamma^*(i^*)} (c_j - c_j^*) \right) = p(C_{\texttt{alg}} - C^*)$$

since $deg(i^*) = p$ for all $i^* \in X^*$. The RHS in the sum is

$$\sum_{(A_i^*, A_i) \in R} \sum_{i' \in A_i} \sum_{j \in \Gamma(i')} 2c_j^* = \sum_{i \in X} deg(i) \sum_{j \in \Gamma(i)} 2c_j^* \leq 2(p+1) \sum_{i \in X} \sum_{j \in \Gamma(i)} c_j^* = 2(p+1)C^*.$$

since $deg(i) \leq p + 1$ for all $i \in X$. Equating the two, $pC_{alg} \leq (3p+2)C^*$. □