IOE 691: Approximation & Online Algorithms

Lecture Notes: Local Search

Instructor: Viswanath Nagarajan Scribe: Qi Luo

A second category of approximation algorithms that we are interested is Local Search (LS). In a greedy algorithm, we are only allowed to do one type of operation (addition or deletion) at each iteration. In LS algorithms, we are able to do both addition and deletion at each iteration. LS algorithms move from one solution to another "neighboring" solution in the search space, by applying local changes. This iterative process continues until no local improvement is possible, ending up with a locally optimal. The analysis will relate the objective of the locally optimal solution to a globally optimal one.

1 Maximum Matching

The maximum matching problem is the following.

Definition 1.1 *Given a graph $G = (V, E)$. a matching is a subset $K \subseteq E$ where every vertex has degree no greater than 1 in K. The goal of the maximum matching problem is to find a matching K with maximum $|K|$.*

For example, in figure 1, the maximum matching value is 4.

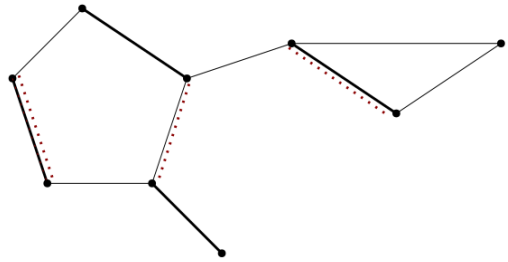

Figure 1: Thick lines are the optimal matching, and red-dotted lines are a feasible matching.

Here we discuss a local search approximation algorithm for maximum matching. (Note that there is even an efficient exact algorithm for matching- using different techniques.)

The LS algorithm is parameterized by a constant $t \geq 1$ which determines the running time and the approximation ratio. Roughly speaking, t is the number of local changes allowed to a solution.

Algorithm 1: Local Search Algorithm for Matching Problem

Data: Graph $G = (V, E)$, with $|V| = n, |E| = m$.
Start with any matching $M \neq \emptyset$
while *there are edges R to remove and A to add such that:*
`/* local move */`
$A \subseteq E$, $R \subseteq M$, $|R| < |A| \leq t$ *and* $(M \backslash R) \cup A$ *is a matching*
do
  $M \leftarrow (M \backslash R) \cup A$

Lemma 1.1 *The running time for the LS algorithm for matching is at most* $n^{O(t)}$.

Proof: Note that the total number of potential local moves is at most $(m^t \times n^t)$; the first term denotes the choices for A and the second term denotes the choices for R. Moreover, the total number of iterations is at most $\frac{n}{2}$: this is because $|M|$ increases by at least 1 in each iteration. So the runtime is at most $O(n^{t+1}m^t)$. Since $m = |E| \leq \binom{n}{2}$, the lemma follows. ■

Theorem 1.1 *The LS algorithm is a* $(1 - \frac{1}{t+1})$*-approximation algorithm for matching.*

Proof: Let N denote the optimal matching for G and M the locally optimal solution output by the algorithm. We consider $M \bigoplus N$ where the exclusive operator $\bigoplus$ denotes the union of edges in M but not N, and edges in N but not M. See Figure 2 for an example.

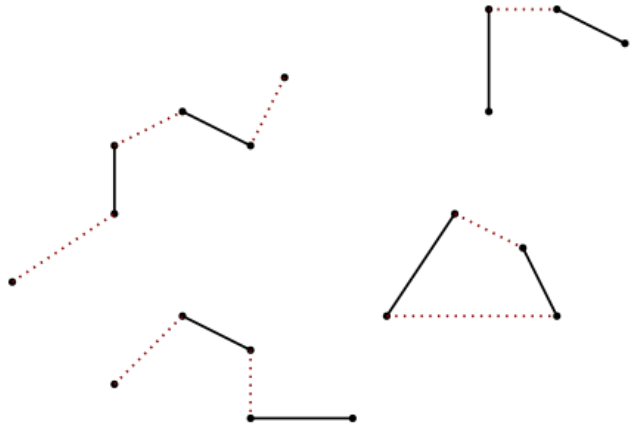

Figure 2: Graph $M \bigoplus N$, where solid lines are M and red-dotted lines are N.

Note that each vertex in $M \bigoplus N$ has degree at most two. So each connected component $M_i \cup N_i$ in $M \bigoplus N$ is of one of the following types:

1. Cycle: then there must be an even number edges and $|M_i| = |N_i|$, otherwise it is contradictory to the definition of matching.

2. Path of even number edges: then $|M_i| = |N_i|$, because edges in either set are disjoint.

3. Path of odd number edges and $|M_i| > |N_i|$: then $|M_i| = |N_i| + 1$ and the approximation output is better than optimal in this subset.

4. Path of odd number edges and $|M_i| < |N_i|$: then $|M_i| = |N_i| - 1$. We observe that we must have $|N_i| \geq t+1$. Suppose that this is not the case: then we can do a swap $R = M_i, A = N_i$ so that $|R| < |A| \leq t$ and $(M \setminus R) \cup A$ is a valid matching (of larger size), which is a contradiction to the local optimality of M. Now, using $|N_i| \geq t + 1$,

$$|M_i| = |N_i| - 1 \geq |N_i| - \frac{|N_i|}{t+1} = (1 - \frac{1}{t+1})|N_i|.$$

Adding over all components $M_i \cup N_i$ we can derive that $|M| \geq (1 - \frac{1}{t+1})|N|$. ■

Note that the running time also depends on t: so there is a trade-off in choosing t for the runtime and approximation ratio.

2 K-median

The k-median problem is a basic facility location problem.

Definition 2.1 *The input is a set V of vertices, distance function d: $V \times V \to \mathbb{R}_+$ (symmetric and satisfies triangle inequality) and a bound k. The goal is to choose a set $S \subseteq V$, $|S| = k$, of cluster centers so as to minimize the sum of distances of each vertex to its closest center in S.*

$$\min_{S \subseteq V, |S|=k} \quad \sum_{j \in V} \min_{u \in S} d(j,u)$$

For any $S \subseteq V$ we will call $Med(S) = \sum_{j \in V} \min_{u \in S} d(j,u) = \sum_{j \in V} d(j,S)$.

Algorithm 2: Local Search Algorithm for k-median problem

Require: Metric (V,d), bound k.
start with any $S \subseteq V$ with $|S| = k$
while there is any pair $a \in V \setminus S, r \in S$ such that
/* local move */
$Med(S - r + a) < Med(S)$ **do**
$S = S - r + a$
end while

We use $(S - r + a)$ as an abbreviation for $(S \setminus \{r\}) \cup \{a\}$.

We need some definitions for the analysis. Let $L = \{l_1, l_2, \cdots, l_k\}$ be the algorithm's solution and $Q = \{q_1, q_2, \cdots, q_k\}$ be the optimal solution. Let $\pi : Q \to L$ be the map defined as $\pi(q_i) =$ closest vertex in L to q_i. Let D_j be the distance between any vertex $j \in V$ to its closest center in L; and D_j^* be the distance between vertex $j \in V$ to its closest center in Q. Let $\{L_1, L_2, \cdots, L_k\}$ denote the partition of V in solution L where each L_i denotes the vertices that are closest to center l_i. Similarly, let $\{Q_1, Q_2, \cdots, Q_k\}$ denote the partition of V in solution q where each Q_i denotes the vertices that are closest to center q_i. See Figure 3. Note that

$$OPT = \sum_{j \in V} D_j^*, \qquad ALG = \sum_{j \in V} D_j.$$

2.1 Special case: π is a bijection

We first bound the approximation ratio when π is a bijection. Without loss of generality, in the following discussion we will have $\pi(q_i) = l_i$ for each $i \in [k]$. See Figure 4

Theorem 2.1 *If π is a bijection, the local search algorithm is a 3-approximation algorithm.*

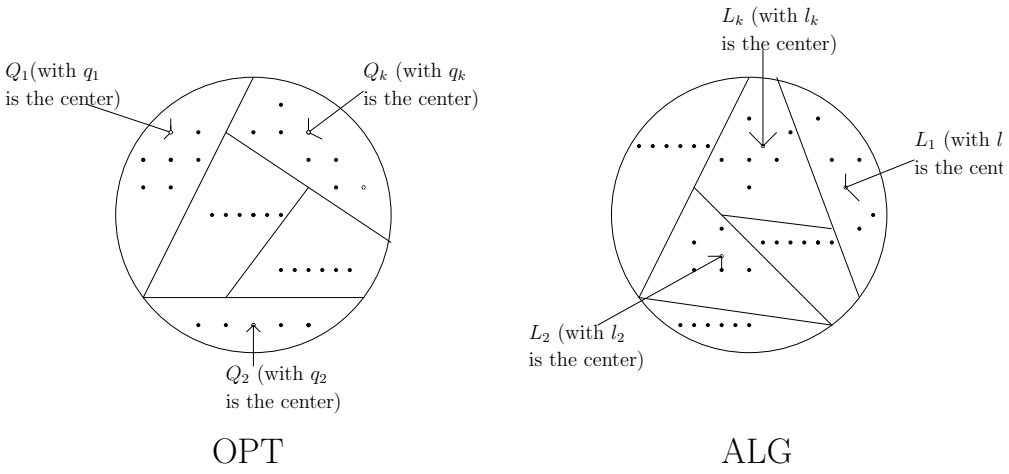

Figure 3: The partition for OPT and ALG.

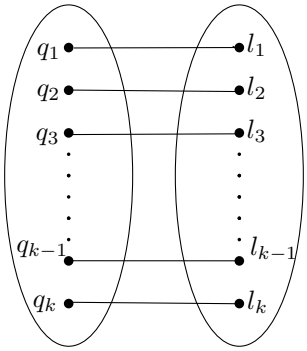

Figure 4: π is a bijection

The following lemma is the key step.

Lemma 2.1 *$Med(L - l_i + q_i) - Med(L) \leq \sum_{j \in Q_i} D_j^* - \sum_{j \in Q_i} D_j + 2 \sum_{j \in L_i} D_j^*$, for any $i \in [k]$.*

Proof: Let $L^{'} = L - l_i + q_i$. We have $Med(L') = \sum_{j \in V} d(j, L^{'})$. We now bound this term by term. The vertices $j \in V$ are divided into three categories:

(i) When $j \in Q_i$, we connect j to q_i in L', i.e. j switches its center from l_i to q_i (see Figure 5). So $d(j, L^{'}) \leq D_j^*$.

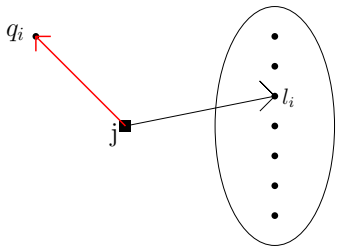

Figure 5: j changes from l_i to q_i (from black arrow to red arrow).

(ii) When $j \in L_i \backslash Q_i$, then j is assigned to $\pi(q_r)$ where q_r represents the center that j is assigned to OPT. See Figure 6. Crucially, $\pi(q_r) \neq l_i$ and so $\pi(q_r) \in L'$. Then we have:

$$\begin{aligned} d(j, \pi(q_r)) &\leq d(j, q_r) + d(q_r, \pi(q_r)) \\ &= D_j^* + d(q_r, \pi(q_r)) \\ &\leq D_j^* + d(q_r, l_i) \qquad (by\ the\ definition\ of\ \pi) \\ &\leq D_j^* + d(q_r, j) + d(j, l_i) \\ &= D_j^* + D_j^* + D_j \end{aligned}$$

So we get that $d(j, L^{'}) \leq d(j, \pi(q_r)) \leq 2D_j^* + D_j$

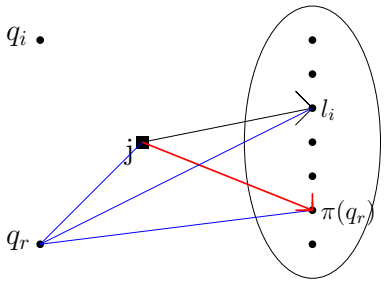

Figure 6: j changes from l_i to $\pi(q_r)$(from black arrow to red arrow).

(iii) When $j \in V \backslash (L_i \cup Q_i)$, it does not change its center, so $d(j, L^{'}) \leq D_j$.

As $-Med(L) = -\sum_{j \in V} D_j$, combining the above three situations together, we get:

$$Med(L - l_i + q_i) - Med(L) \leq \sum_{j \in Q_i} D_j^* - \sum_{j \in Q_i} D_j + 2 \sum_{j \in L_i} D_j^*$$

So we finished proving the lemma. ■

Now we will prove Theorem 2.1. By adding Lemma 2.1 for all $i \in [k]$, the LHS will be $\sum_{i \in [k]} Med(L - l_i + q_i) - Med(L)$. By the definition of Local Search Algorithm, we know that $Med(L - l_i + q_i) - Med(L) \geq 0$ for all $i \in [k]$ as L is locally optimal. This means $\sum_{i \in [k]} Med(L - l_i + q_i) - Med(L) \geq 0$. For the RHS, it will become

$$\begin{aligned} \sum_{i \in [k]} (\sum_{j \in Q_i} D_j^* - \sum_{j \in Q_i} D_j + 2 \sum_{j \in L_i} D_j^*) &= \sum_{i \in [k]} \sum_{j \in Q_i} D_j^* - \sum_{i \in [k]} \sum_{j \in Q_i} D_j + 2 \sum_{i \in [k]} \sum_{j \in L_i} D_j^* \\ &= OPT - ALG - 2OPT \end{aligned}$$

So we have $0 \leq OPT - ALG + 2OPT$, which implies that $ALG \leq 3OPT$.

2.2 General π

Now we turn to general case when π is not necessarily a bijection.

Theorem 2.2 *The 1-swap local search algorithm for k-median is a 5-approximation algorithm.*

First of all, we will define the candidate swaps. Recall that $\pi : Q \to L$. Define $U \subseteq Q$ to consist of those vertices $q \in Q$ which are not uniquely mapped to $\pi(q)$, i.e.

$$U = \{q \in Q : \exists q' \in Q \setminus q \text{ with } \pi(q) = \pi(q')\}.$$

Define $Z \subseteq L$ to consist of those $l \in L$ that are not in the image of π, i.e.

$$L = \{l \in L : |\pi^{-1}(l)| = 0\}.$$

As $|U|$ needs to be at least $2(|U| - |Z|)$ by its definition, we can easily get that $|Z| \geq \frac{|U|}{2}$. Then we introduce the following candidate swaps (see Figure 5). There is one swap for each $q \in Q$:

(1) If $\pi(q)$ is unique to q (i.e. $q \in Q \setminus U$), then let the swap add q and remove $\pi(q)$. This is identical to the bijection special case.

(2) For $q \in U$, pair them arbitrary with elements in Z such that:

   (i) each $q \in U$ is added in one pair;

   (ii) each $l \in Z$ is removed in at most two pairs.

   Note that this is possible as $|U| \leq 2|Z|$.

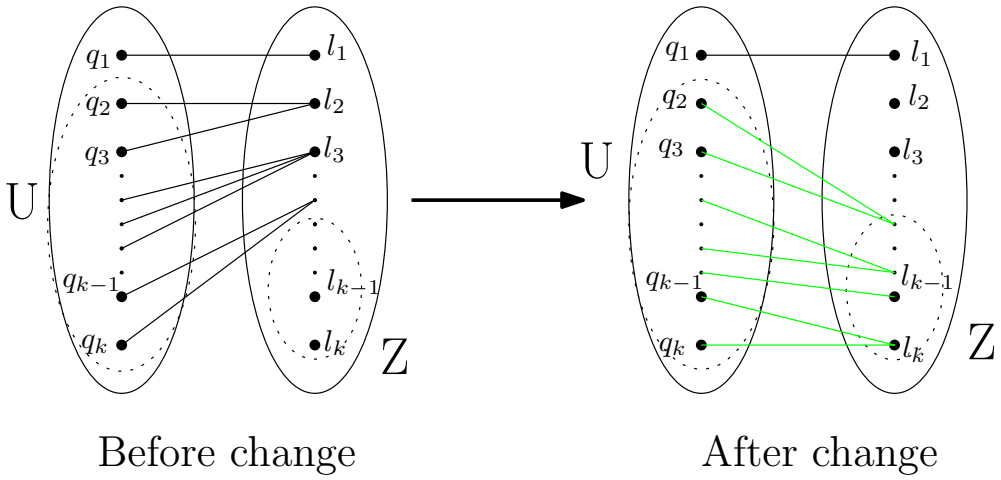

Figure 7: candidate swap

Let Y denote the set of pairs above; note that each is of the form (l, q) where $l \in L$ is removed and $q \in Q$ is added.

Lemma 2.2 *For any $(l, q) \in Y$, $Med(L - l + q) - Med(L) \leq \sum_{j \in Q_0} D_j^* - \sum_{j \in Q_0} D_j + 2 \sum_{j \in L_0} D_j^*$ where Q_0 and L_0 are the parts in $\{Q_i\}_{i=1}^k$ and $\{L_i\}_{i=1}^k$ corresponding to q and l.*

Proof: The proof is almost identical to Lemma 1.1. Let $L' = L - l + q$. For $j \in Q_0$, the situation doesn't change, so we also have $d(j, L') \leq D_j^*$. For $j \in L_0 \setminus Q_0$, we still can pick $\pi(q_r)$ as the new center for j: (i) if $q \in Q \setminus U$ then this is identical to the case in Lemma 1.1, and (ii) if $q \in U$ then we know that $l \in Z$ (i.e. $\pi^{-1}(l) = \emptyset$), so $\pi(q_r) \neq l$ and we can indeed assign j to $\pi(q_r) \in L'$. So $d(j, L') \leq d(j, \pi(q_r) \leq 2D_j^* + D_j$ still holds for $j \in L_0 \setminus Q_0$. For $j \in V \setminus (L_0 \cup Q_0)$, it is the same as before, so we have $d(j, L') \leq D_j$. As $-Med(L) = -\sum_{j \in V} D_j$, combining the above three situations together, we get:

$$Med(L-l+q)-Med(L) \leq \sum_{j\in Q_0} D_j^* - \sum_{j\in Q_0} D_j + 2\sum_{j\in L_0} D_j^*$$

So we finished proving the lemma. ■

Now we will finish the proof of Theorem 2.2. By adding Lemma 2.2 for all $i \in [k]$, the LHS will be $\sum_{(l,q)\in Y} Med(L-l+q) - Med(L) \geq 0$. For the RHS, the first two items will still be $OPT - ALG$. But the third item will be different. As each center $l_i \in L$ occurs in 0,1 or 2 swaps, we can give an upper bound by double the summation $2\sum_{i\in[k]}(\sum_{j\in L_i} D_j^*) \leq 4\sum_{j\in V} D_j^* = 4OPT$. So we have $0 \leq OPT - ALG + 4OPT$, i.e. $ALG \leq 5OPT$.

2.3 Polynomial-time algorithm

The running time of the above local-search may depend polynomially on the distances in d, which is not adequate for a polynomial-time algorithm. Recall that the input size is polynomial in n and $\log D$ where D is the maximum distance in d (assuming all integer distances). We can ensure a polynomial runtime by making a local move only when Med decreases by an ϵ-factor. So each swap will satisfy $Med(S') - Med(S) < -\epsilon Med(S)$, where S is the original set and S' is the set after swap. If we start with solution S_0 and end with solution S_k after k swaps, then

$$1 \leq Med(S_k) \leq (1-\epsilon)^k \cdot Med(S_0) \leq (1-\epsilon)^k \cdot nD.$$

This bounds the number of iterations by $O(\frac{1}{\epsilon}\log(nD))$ which is polynomial. This also does not affect the approximation ratio much. For any L' obtained from a swap of L, we know $-\epsilon \cdot Med(L) \leq Med(L') - Med(L)$. As we only added n such swaps in the analysis, we will get a $5/(1-n\epsilon)$ approximation ratio. Setting $\epsilon = \frac{1}{n^2}$ gives:

Theorem 2.3 *The local search algorithm for the k-median problem that uses bigger improving swaps yields a* $5 + o(1)$ *approximation algorithm.*

2.4 Tight Example

We now show that the 1-swap local search algorithm for k-Median has a tight approximation ratio of 5. That is, there is some instance where the locally optimal solution costs at least 5 times the optimum. Consider the example in Figure 8. The black circles denote the "clients" whereas the white squares and circles denote possible centers. The bottom row is a local optimum and has cost equal to $2(k-1)+\frac{k+1}{2}$. It is easy to check that exchanging one center from the local optimum with another center, does not lead to any reduction in cost. The top row is the optimal solution with cost $\frac{k+1}{2}$. So the ratio of the algorithm's cost to the optimum is at least $5 - o(1)$ as k grows large.

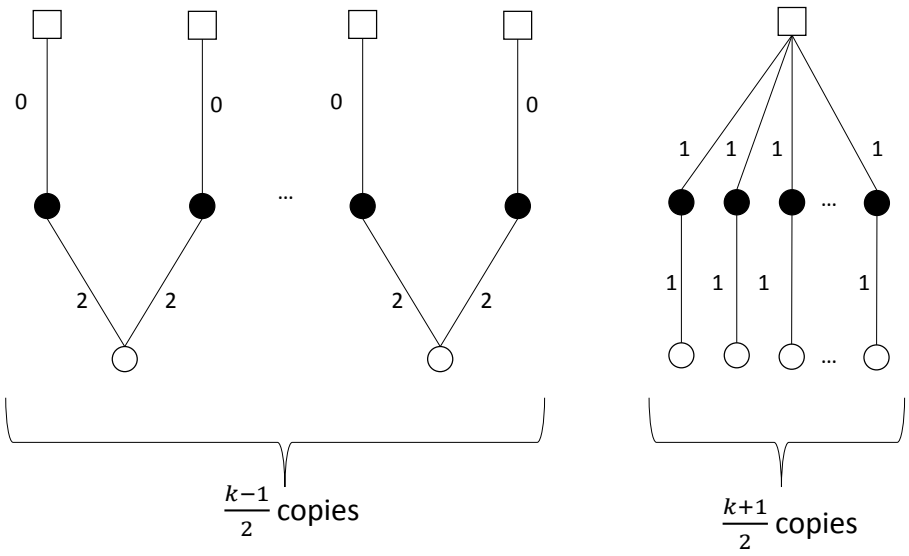

Figure 8: Tight example for 5-approximation of local search algorithm in k-median problem

The local search algorithm can also be modified to swap up to t centers at any step. The paper [AGK$^+$04] shows that the t-swap local search for is a $\left(3+\frac{2}{t}\right)$-approximation. Using more recent LP-based techniques a better ≈ 2.67 approximation is known for k-Median [BPR$^+$17]. It is also known that one cannot obtain an approximation better than ≈ 1.73 for k-Median [JMS02].

References

[AGK$^+$04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[BPR$^+$17] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017.

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740, 2002.