15-854: Approximations Algorithms | **Lecturer:** Anupam Gupta
Topic: Local Search: Max Cut Revisited and k-Median | **Date:** 10/3/05
Scribe: Mohit Singh

7.1 Introduction

In Lecture 2, we discussed the unweighted max cut problem and gave 2-approximation algorithms based on local search, greedy paradigm and randomization. Today, we will study the weighted version of the max cut problem. Using this problem we will illustrate a general technique of bounding the running time of a local search based algorithm at the expense of a small factor in the approximation guarantee. We will then move on to the k-median problem and show a natural local search algorithm gives a 5-approximation.

7.2 Weighted Max Cut

The weighted max-cut(WMC) problem is defined as follows: given a graph $G = (V, E)$, a weight function $w : E \rightarrow \mathcal{Z}_+$, find a set $S \subset V$ such that the $w(S, \bar{S})$ is maximum. The WMC problem is NP-hard.(See Lecture 2 for details).

A natural local search algorithm starts with some initial cut $(S, \bar{S})$ and tries to move a vertex from S to $\bar{S}$ or vice-versa if it increases the value of the cut. This algorithm in general may take as much $O(W)$ time, where $W = \sum_{e\in E} w_e$ which need not be polynomial bounded by the size of problem instance.

The following modified local search algorithm **Mod-Local-Search** bypasses this problem by only making *significant* improvements. The algorithm also needs as an parameter $\epsilon > 0$ and returns a $2 + \epsilon$-approximate solution. Running time of the algorithm is inversely related to ϵ.

Algorithm **Mod-Local-Search**:

1. *(Initialize)* Let $S \leftarrow \{v\}$ where $w(\delta_v) = max_{u\in V} w(\delta(u))$. Here, $\delta(v) = \{e \in E | e = (u, v)$ for some $u \in V\}$
2. *(Improvement)* While $\exists v \in \bar{S}$ such that $w(S \cup \{v\}, \bar{S} \setminus \{v\}) \geq \epsilon \cdot \frac{w(S,\bar{S})}{n}$ (Condition 1) or there exists a $v \in S$ such that $w(S \setminus \{v\}, \bar{S} \cup \{v\}) \geq \epsilon \cdot \frac{w(S,\bar{S})}{n}$ (Condition 2),
   (a) If Condition 1 holds then let $S \leftarrow S \cup \{v\}$
   (b) If Condition 2 holds then let $S \leftarrow S \setminus \{v\}$
3. Return S

We prove the following theorem about the algorithm **Mod-Local-Search**.

Theorem 7.2.1 *Given any fixed $\epsilon > 0$, algorithm **Mod-Local-Search** returns a solution S such that $w(S,\bar{S}) \leq \frac{1}{2+\epsilon} w(O,\bar{O})$ where O is the optimal solution to the max-cut problem. Moreover, the algorithm runs in time $O(\frac{n^2 \log n}{\epsilon})$ where n is the order of G.*

Proof: Let S be the solution returned by the algorithm and let $W = \sum_{e \in E} w_e$. As S is a *near* local optimum solution we have the following inequalities. For each $v \in S$,

$$w(S,\bar{S}) \geq w(S \setminus v, \bar{S} \cup v) - \epsilon \frac{w(S,\bar{S})}{n}$$

$$\Rightarrow w(v,\bar{S}) \geq w(v,S) - \epsilon \frac{w(S,\bar{S})}{n} \tag{7.2.1}$$

Also, for each $v \notin S$,

$$w(S,\bar{S}) \geq w(S \cup v, \bar{S} \setminus v) - \epsilon \frac{w(S,\bar{S})}{n}$$

$$\Rightarrow w(v,S) \geq w(v,\bar{S}) - \epsilon \frac{w(S,\bar{S})}{n} \tag{7.2.2}$$

Adding the above n inequalities 7.2.1, 7.2.2, we have

$$2w(S,\bar{S}) \geq W - \epsilon w(S,\bar{S})$$

$$\Rightarrow w(S,\bar{S}) \geq \frac{W}{2+\epsilon}$$

The claim in the theorem follows as $w(O,\bar{O}) \leq W$ for any $O \subset V$.

Now, we show that there are $O(\frac{n \log n}{\epsilon})$ iteration of Step *Improvement*. Let $S_0, \ldots, S_r = S$ be the sets encountered in the running of the algorithm. Then $w(S_0,\bar{S_0}) \geq \frac{W}{n}$. Also

$$\begin{array}{rlcl} & w(S_i,\bar{S_i}) & \geq & (1+\frac{\epsilon}{n}) w(S_{i-1},\bar{S}_{i-1}) \\ \Rightarrow & w(S_r,\bar{S_r}) & \geq & (1+\frac{\epsilon}{n})^r w(S_0,\bar{S_0}) \\ \Rightarrow & r & \leq & \frac{\log \frac{w(S_r,\bar{S_r})}{w(S_0,\bar{S_0})}}{\log(1+\frac{\epsilon}{n})} \end{array}$$

Now, using $w(S_r,\bar{S_r}) \leq W$ and $w(S_0,\bar{S_0}) \geq \frac{W}{n}$, $\log(1+x) \geq \frac{x}{2}$ for small enough x, we have $r \leq \frac{2n \log n}{\epsilon}$. As each iteration can be implemented in $O(n)$ time, we have the claim in theorem. ∎

Observe that in the above algorithm we did not reach some local optimum but stopped at a near local optimum solution. Usually, near local optimality is enough to prove guarantees exactly the same guarantee as local optimum solution. This gives the added advantage as near local optimum solutions are faster to find than a local optimum.

Exercise 7.2.2 *Can you think of a polynomial time 2-approximation for the weighted max cut beating the above $2+\epsilon$-approximation?*

7.2.1 Another Local Search Heuristic for Max-Cut

For a local search heuristic for any problem, we need a notion of neighborhood solutions to any feasible solutions. The local search heuristic starts from any feasible solution S_1 and selects an improving solution $S_2 \in N(S_1)$ where $N(S_1)$ denotes the neighboring solutions of S_1. The selection rule for improving solution might involve selecting the best solution in $N(S_1)$ or any improving solution or any significantly improving solution.

For the max-cut problem, we defined a neighborhood solutions $N(S)$ for any solution in a very natural manner. Another neighborhood criterion was given by Lin and Kernighan [?] which works very well in practice for a large class of graph partitioning problems including max-cut. For the max-cut problem, the neighborhood set for any cut $(S, \bar{S})$ is defined as follows:

1. Unmark each vertex. Let $S_0 = S$
2. For $i = 1$ to n,
   (a) Select an unmarked node v_i such that *shifting* v_i in the solution $(S_{i-1}, \bar{S}_{i-1})$ gives the best solution over all shifting of unmarked nodes.
   (b) Mark v_i and shift v_i in $(S_{i-1}, \bar{S}_{i-1})$ to obtain the solution $(S_i, \bar{S}_i)$.
3. Return $N(S) = \{S_0, \dots, S_n\}$

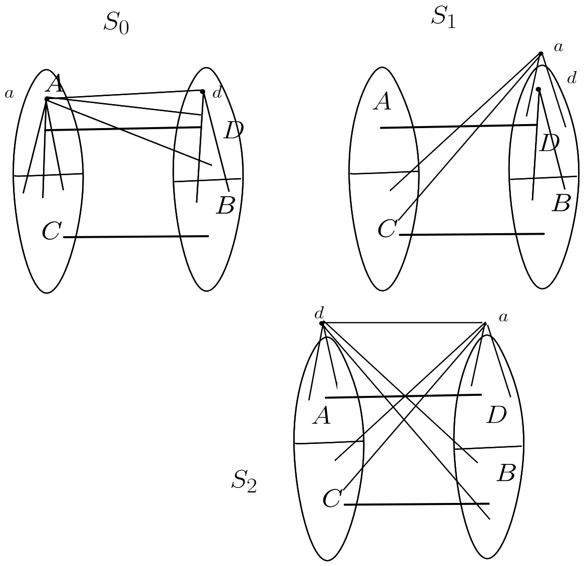

Figure 7.2.1: First two steps of Kernighan-Lin heuristic for maxcut. $w(S_0, \bar{S_0}) = 2n^2 = w(S_1, \bar{S_1})$. But $w(S_2, \bar{S_2}) = n^2 + 1$

Observe that S_n corresponds to the same solution as $S_0 = S$. The local search algorithm then selects the best of these solutions in a single iteration and continues till it reaches a local optimum.

To illustrate the power that we do not get *stuck* in a bad local optimum solution which happened in the case of the **Mod-Local-Search**. Consider the complete bipartite graph $G = (V, E)$ with bipartition $(A \cup B, C \cup D)$ where $|A| = |B| = |C| = |D| = n$. We saw in Lecture 2, that $S = (A \cup C)$ defines a local optimum solution for **Mod-Local-Search** with value about half the optimum solution. We now show that the corresponding to $S = A \cup C$ is not a local optimum under the Kernighan-Lin neighbourhood criterion and local search algorithm based on this criterion will still make progress.

While defining $N(S)$, in the first step we will shift arbitrarily any vertex from one side to another. Suppose, wlog, we shift $v \in A$ from S to $\bar{S}$. Hence, $S_1 = A \cup C \setminus v$. Now observe that we will shift some vertex of D on the other side as shifting any vertex from D increases the value of the cut by at least 1. And already we assured of making progress as $c(S_2, \bar{S_2}) > c(S, \bar{S})$.

Unfortunately, the Kernighan-Lin heuristic has not proven to give performance guarantees for any natural family of graphs.

7.3 k-Median

Given a complete graph $G = (V, E)$ with edge weights d_e on edge e which satisfy the triangle inequality and an integer $k \leq n$, the k-median problem asks for a set $S \subset V$ with $|S| = k$ such that $cost(S) = \sum_{v \in V} d(v, S)$ is minimized. Here, $d(v, S) = min_{u \in S} d(u, v)$.

While the k-center problem, which we have seen in lecture 2 and assignment 1, is used to minimize the maximum distance of any vertex to set S, in the k-median problem we try to minimize the average distance of all vertices to S. We will also refer S to set of open facilities.

We will analyze a very natural local search algorithm for facility location.

Alg-LS:

1. Start with any $S \subset V$ such that $|S| = k$.
2. While there exists $s \in S$ and $x \notin S$ such that $cost(S \cup x \setminus s) < cost(S)$
   - $S \leftarrow S \cup x \setminus s$.
3. return S

Observe that as in weighted max-cut the above algorithm is not guaranteed to stop in polynomial time at a local optimum solution. But, we can modify the local search algorithm to near local search algorithm by changing the current solution only if we reduce the cost by a significant amount. This will ensure that the algorithm ends in polynomial time and the algorithm ends with a near local optimum solution. The analysis of this argument is identical to one in the second case.

Now, we will analyze the cost of a local optimum solution. The cost of a near optimum solution can be bounded by the same argument losing an extra ϵ in the approximation guarantee.

We will prove the following theorem from Arya et al [1] about the cost of any locally optimum solution which will show the performance guarantee of solution returned by the algorithm **Alg-LS**.

Theorem 7.3.1 ([1]) *Any local optimum solution $\mathcal{A} \subset \mathcal{V}$ to the k-median problem has cost at most $5 \cdot cost(\mathcal{B})$ where $\mathcal{B} \subset \mathcal{V}$ is the optimal solution.*

Proof: First, we introduce some notation which we will use in the proof of the algorithm. Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ and $\mathcal{B} = \{b_1, \ldots, b_k\}$. For any $v \in V$, let $A_v = d(v, \mathcal{A})$ and $B_v = d(v, \mathcal{B})$. For each $a \in A$, let $N_A(a) = \{v | d(v, A) = d(v, a)\}$, i.e., vertices that are assigned to a in the k-median solution $\mathcal{A}$. If a vertex x can be assigned to two or more facilities, then we assign it arbitrarily to any *one*. Similarly, we define $N_B(b)$ to the vertices assigned to b in solution $\mathcal{B}$. Also, for any set $X \subset V$ and $v \in V$, we denote $X + v = X \cup \{v\}$ and $X - v = X \setminus \{v\}$. Clearly,

$$cost(\mathcal{A}) = \sum_{v \in V} A_v \tag{7.3.3}$$

$$cost(\mathcal{B}) = \sum_{v \in V} B_v \tag{7.3.4}$$

Local optimality gives us the following condition

$$cost(\mathcal{A}+x-a) \;\geq\; cost(\mathcal{A}) \qquad\qquad \forall x \notin \mathcal{A},\; a \in \mathcal{A} \tag{7.3.5}$$

A pair (a, x) for which the inequality 7.3.5 is applicable is called a swap pair. Although the local optimality condition holds for each $x \notin \mathcal{A}$ and $a \in \mathcal{A}$, we will apply it to special set of pairs. We introduce the notion of capture before we give the special set of pairs. Given a facility $a \in \mathcal{A}$, and a facility $b \in \mathcal{B}$, we say a captures b if $|N_A(a) \cap N_B(b)| > \frac{1}{2}|N_B(b)|$.

Observation 1 *No two different facilities in $\mathcal{A}$ can capture the same facility in $\mathcal{B}$ while a single facility in $\mathcal{A}$ can capture many facilities in $\mathcal{B}$.*

We will select a set of k swaps of form (a_i, b_j) to apply inequality 7.3.5 which satisfy the following conditions:

1. Each $b \in \mathcal{B}$ occurs in exactly one swap pair.
2. Each $a \in \mathcal{A}$ occurs in at most two swap pairs.
3. If (a, b) occurs in one of the k selected swap pair then a does not capture any facility in $\mathcal{B} - b$.

We find the swaps by following procedure. For each $a \in \mathcal{A}$ which captures exactly one facility $b \in \mathcal{B}$, we pair (a, b) and remove both a and b from the set of candidates for future set of swaps possible. Let there be $x \leq k$ pairs matched by the above method. Let A^0 denote the set of facilities in $\mathcal{A}$ which do not capture any facility in $\mathcal{B}$.

Claim 7.3.2 $|A^0| \geq \frac{k-x}{2}$

Proof: Each facility in $\mathcal{B}$ can be captured by at most 1 facility in $\mathcal{A}$. There are exactly x facilities in $\mathcal{A}$ which capture exactly 1 facility of $\mathcal{B}$. Let there be t facilities in $\mathcal{A}$ which capture at least 2 facilities of $\mathcal{B}$. Then we have

$$\begin{array}{rlcl} & x + 2t & \leq & k \\ & x + t + |A^0| & = & k \\ \Rightarrow & |A^0| & \geq & \frac{k-x}{2} \end{array}$$

■

There are $k - x$ candidate facilities in $\mathcal{B}$ left to paired with some facility in $\mathcal{A}$ in some swap pair. We pick any $\frac{k-x}{2}$ facilities of A^0 and pair each of them with two remaining facilities of $\mathcal{B}$. Observe that this swap pairing satisfies all three conditions mentioned above. Note that, we have not paired every facility in $\mathcal{A}$ but every facility of $\mathcal{B}$ has been paired with some facility in $\mathcal{A}$.

Now, consider any of the selected swap pair (a, b). The inequality 7.3.5 implies that

$$cost(\mathcal{A} + b - a) \geq cost(\mathcal{A})$$

where $cost(\mathcal{A} + b - a)$ is calculated according to optimal assignment of vertices to their nearest facilities in $\mathcal{A} + b - a$. Instead, we will give a different assignment of each vertex to facilities in $\mathcal{A} + b - a$. Clearly, the value of the objective function, which we denote by $\hat{cost}(A + b - a) \geq cost(\mathcal{A} + b - a)$.

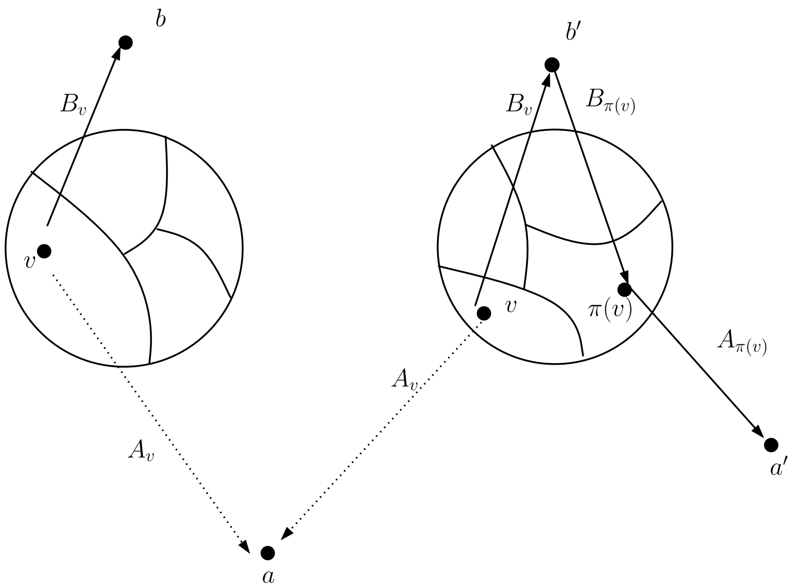

Figure 7.3.2: The new assignments are shown in dark lines and old assignment in dotted lines for the both the cases.

The assignment is as follows: each vertex $v \in N_B(b)$ is assigned to b. Each vertex $v \in N_A(a) \setminus N_B(b)$ lies in $N_B(b')$ for some $b' \in B - b$. We assign v to a' such that $\pi(v) \in N_A(a')$ where π is a permutation we will define later. (We will show later that $a' \neq a$ and such an assignment is meaningful). Notice, that this assignment cost is at most $B_v + B_{\pi(v)} + A_{\pi(v)}$ as can be verified from Figure 7.3.2. Every other vertex $v \in V \setminus (N_B(b) \cup N_A(a))$ is assigned to same facility as in assignment $\mathcal{A}$.

The local optimality condition gives us that

$$\hat{cost}(\mathcal{A} + b - a) - cost(\mathcal{A}) \geq 0$$
$$\Rightarrow \sum_{v \in N_B(b)} (B_v - A_v) + \sum_{v \in N_A(a) \setminus N_B(b)} (B_v + B_{\pi(v)} + A_{\pi(v)} - A_v) \geq 0 \qquad (7.3.6)$$

The second inequality follows as rest of the terms are exactly the same in the two assignments and cancel each other. We will come back to the above inequality which holds for each of the k selected swap pairs. Before that, we take a digression to define the permutation π.

Digression: Defining Permutation π We will define the permutation over each of the sets $N_B(b)$ for each b separately. As these sets are disjoint and partition V, the permutation π will be well defined. For any $b \in \mathcal{B}$, arrange the vertices of $N_B(b)$ in a cyclic order such that vertices in $N_A(a)$ occur consecutively for any $a \in \mathcal{A}$. For any vertex $v \in N_B(b)$, π maps the vertex v to its diametrically opposite vertex in the cyclic ordering.

Claim 7.3.3 *If a does not capture b, then for any vertex $v \in N_A(a) \cap N_B(b)$ we have $\pi(v) \notin N_A(a)$.*

Proof: If $\pi(v) \in N_A(a)$, then every vertex between v and $\pi(v)$ is also in $N_A(a)$. Hence, $N_A(a) \cup N_B(b) > \frac{|N_B(b)|}{2}$ as v and $\pi(v)$ were diameterically opposite points in the cyclical ordering. But, that means a captures b. A contradiction. ■

The above property shows that the assignment considered when applying the local optimality condition is meaningful. Any selected swap pair (a, b) has the condition that a cannot capture any other $b' \in \mathcal{B} - b$. Hence, using Claim 7.3.3 we have that each $v \in N_A(a) \setminus N_B(b)$ is assigned to some facility in $\mathcal{A} - a$ and not back to a itself which is no longer a facility in $\mathcal{A} - a + b$.

If we sum inequality 7.3.6 over each of the k swap pairs, we obtain that the first term is exactly

$$\sum_{(a,b)\in P} \sum_{v \in N_B(b)} (B_v - A_v) = \sum_{v \in V} B_v - A_v = cost(\mathcal{B}) - cost(\mathcal{A}) \tag{7.3.7}$$

where the equality follows from the the fact that each b occurs exactly once among all the pairs and $N_B(b)$ for different b partition V.

Observe that $B_v + B_{\pi(v)} + A_{\pi(v)} \geq A_v$ for any $v \in V$ as LHS is more than the distance of v to some facility in $\mathcal{A}$ and RHS is the distance of v to its nearest facility in $\mathcal{A}$. Hence, we can simplify the second summation to

$$\begin{aligned}
\sum_{(a,b)\in P,\ v \in N_A(a)\setminus N_B(b)} (B_v + B_{\pi(v)} + A_{\pi(v)} - A_v) &\leq \sum_{(a,b)\in P,\ v \in N_A(a)} (B_v + B_{\pi(v)} + A_{\pi(v)} - A_v) \\
&\leq 2\sum_{v \in V} (B_v + B_{\pi(v)} + A_{\pi(v)} - A_v) \\
&= 2cost(\mathcal{B}) + 2cost(\mathcal{B}) + 2cost(\mathcal{A}) - 2cost(\mathcal{A}) \\
&= 4cost(\mathcal{B})
\end{aligned}$$

where the last inequality follows as each $a \in \mathcal{A}$ occurs at most twice among all pairs and the equality follows as π is a permutation. Combining we obtain

$$5cost(\mathcal{B}) - cost(\mathcal{A}) \geq 0$$

As $\mathcal{B}$ was the optimal solution, this proves the theorem.

■

Arya et al [1] also prove that if instead of swapping one vertex of $\mathcal{A}$ for one vertex not in $\mathcal{A}$, we swapped $p \geq 1$ vertices in each local step, we obtain a $3 + \frac{2}{p}$-approximation for the k-median problem.

References

[1] Vijay Arya, Naveen Garg, Rohit Khandekar, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *ACM Symposium on Theory of Computing*, pages 21–29, 2001.