CS880: Approximation and Online Algorithms **Scribe:** Mikhail Nedbai

Lecture 4: Local Search **Date:** 09.11.2019

4.1 Introduction

Local search algorithms belong to a class of algorithms, which employ heuristc techniques in approximately solving optimization problems. In general, a local search algorithm starts from an arbitrary feasible solution to the problem, and makes incremental progress in modifying the solution, while maintaining feasibility, until no modification to the current solution can imporve the cost. While this techniqe is widely used in practice, it suffers from several drawbacks :

- A local search algorithm that terminates only returns a localy optimal solution, which could vary in cost drastically compared to the global optimum.
- In many cases finding even a local optimum within a constrained number of iterations can be challenging.
- Since most local search algorithms are heuristic, they tend to give minial theoretical guatantees.

There are of course ways to deal with the above mentioned issues. As we will see below, utilizing appropriate local structure will allow us to provide guarantees on both quality of locally optimal solutions in respect to the global optimum, as well as the time complexity of the algorithm's convergence.

4.2 Max Cut Local Search Algorithm

4.2.1 Problem statement

We now present a local search algorithm for the Max Cut problem. We first define a cut in a graph

Definition 4.2.1 *(Cut of a graph) Let $G = (V, E)$ be a graph and let $w_{i,j}, (i, j) \in E$ be the weight of each edge in the graph. Then a partition of the vertex set $(S, V \setminus S)$ is a cut in the graph. Cost of the cut C is defined as $C(S) = \sum_{(i,j)\in E\cap S\times(V\setminus S)} w_{i,j}, \forall S \subseteq V$*

The Max cut problem is defined as follows :

Definition 4.2.2 *(Max Cut) Let $G = (V, E)$ be a graph with positive edge weights $w_{i,j}$, then the Max cut of a graph is $S^* = \max_{S\subseteq V} C(S)$.*

Max cut problem is NP hard, and further more is APX-Hard for any approximation factor better than $\frac{17}{16}$. An algorithm that utilizes semidefinite programming is shown to achieve approximation ratio very close to this value. Here we describe a simple 2 approximation local search algorithm for max cut. For this version of the algorithm, we will allow two modifications to be made on each iteration, $S \cap \{v\}$ and $S \setminus \{v\}$, since a feasible solution to Max Cut is any $S \subseteq V$. Other local search algorithms could incorporate different modifications, such as exchanging verticies between the two sides of the cut.

4.2.2 Algorithm

With that, the algorithm is defined as follows :

- Start $S \leftarrow \emptyset$
- while $\exists v : C(S \cup \{v\}) > C(S) \vee C(S \setminus \{v\}) > C(S)$
  - perform either $S \leftarrow S \cup \{v\}$ or $S \leftarrow S \setminus \{v\}$ to increase the cost of the cut.

4.2.3 Analysis

Claim 4.2.3 *The algorithm above achieves a 2 approximation for Max Cut*

Proof: Suppose a local optimum S is reached. Then consider any $i \in S$. Local optimality of S implies that

$$\sum_{j \in V \setminus S} w_{i,j} \geq \sum_{j \in S} w_{i,j} \tag{4.2.1}$$

$$2 \sum_{j \in V \setminus S} w_{i,j} \geq \sum_{j \in V} w_{i,j} \tag{4.2.2}$$

Similarly consider any $i \in V \setminus S$. Local optimality of S entails

$$\sum_{j \in S} w_{i,j} \geq \sum_{j \in V \setminus S} w_{i,j} \tag{4.2.3}$$

$$2 \sum_{j \in S} w_{i,j} \geq \sum_{j \in V} w_{i,j} \tag{4.2.4}$$

Then summing up over all $i \in V$

$$\begin{aligned} 2C(S) &= \sum_{i \in S} \sum_{j \in V \setminus S} w_{i,j} + \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j} && (4.2.5) \\ &\geq \sum_{i \in S} \frac{1}{2} \sum_{j \in V} w_{i,j} + \sum_{i \in V \setminus S} \frac{1}{2} \sum_{j \in V} w_{i,j} && (4.2.6) \\ &= \sum_{e \in E} w_e && (4.2.7) \end{aligned}$$

It is trivial to observe that OPT cannot be larger that the total weight of all edges in the graph, and thus $2C(S) \geq OPT$, and the claim is proven. ■

Now that we have prooved that this is a 2 - approximation algorithm, we exmaine it's runtime. At each step the algorithm considers 2 actions for every vertex, addition or removal, both of which take time $O(|V|)$, and thus performs $O(|V|^2)$ operations on each iteration. If the weights are assumed to be integral, then the maximum number of iterations the algorithm makes is $\sum_{e\in E} w_e$, since the cost at each iteration increases by at least 1, and the maximum cost that can be achieved is the sum of weights of all edges. The total number of iterations is thus $O(|V|^2 \sum_{e\in E} w_e)$. If the edge weights are unbounded, running time of this algorithm thus becomes non - polynomial, however this can be mititgated by requiring that sufficient progress is made on each iteration. If we mandate that the cost of the cut increases by $(1+\epsilon), \epsilon > 0$ on each iteration, we obtain an algorithm with an approximation ratio of $2+\epsilon$.

4.3 Facility location problem

4.3.1 Problem statement

The Facility Location problem concerns building a set of facilities that would serve a set of customers. Opening each facility incurrs a cost, and additionally each client incurrs a cost of traveling to their nearest facility. In the version of the problem that we will concider, the Metric Uncapacitated Facility Location problem, any number of facilities can be open, and the costs incurred by clients are proportional to a distance metric. More formally :

Definition 4.3.1 *(Metric)*
A metric on set M is a function $d : M \times M$ such that

- $d(x,y) = d(y,x)$
- $d(x,z) \le d(x,y) + d(y,z)$
- $d(x,x) = 0$

One way to think about the facility location problem is to define it on a a graph. Then formally :

Definition 4.3.2 *(Facility Location)*
Let $X, Y \subseteq V$ denote the set of facilities and the set of customers respectively. Let f_i denote the cost of opening location i, and let $c_{i,j}$ denote the cost of connecting client j to facility i. Then the objective is

$$\min_{S\subseteq X, S\neq\emptyset} \sum_{i\in S} f_i + \sum_{j\in Y} \min_{i\in S} c_{i,j}$$

Here we will show a constant factor approximation algorithm for this problem. The assumption of a distance metric on the costs of connecting clients to facilities will allow us to do so, since a more general version of this problem without this assumption only allows for a $O(log(n))$ approximation factor for the problem. More specifically, we will present an algorithm that achieves a 3 approximation to this problem, although the analysis we provide further on will show a 5 approximation factor. More generally the best algorithm for this problem to date achieves an approximation ratio of roughly 1.52 which is very close to the theoretical lower bound of 1.46, approximating beyound which is NP-Hard.

4.3.2 Algorithm

- Start with arbitrary $S \subseteq X, S \neq \emptyset$
- Carry out deletions, insertions and swaps of facilities in X, as long as $C(S)$ lowers.
- retunr the localy optinal S.

4.3.3 Analysis

Theorem 4.3.3 *The above algorithm approximates Facility Location problem with a factor of* 5, *given that a localy optimal solution is returned. Formally* $C(S) \leq 5C(S^*)$, *where* S^* *is the optimal solution to the problem.*

In order to prove the above claim, we first prove several lemmas. We introduce several shorthands for notation :

- $r_j^S = \min_{i \in S} c_{i,j}$
- $C_r(S) = \sum_{j \in Y} r_j^S$
- $C_f(j) = \sum_{i \in S} f_i$
- $C(S) = C_f(S) + C_r(S)$
- $\sigma(j)$ is the facility to which j is assinged in solution S
- $\sigma^*(j)$ is the facility to which j is assinged in solution S*
- $R_i = \sum_{j:\sigma(j)=i} r_j$
- $R_i^* = \sum_{j:\sigma(j)=i} r_j^*$

The first lemma will help us bound $C_r(S)$

Lemma 4.3.4 $C_r(S) \leq C(S^*)$

Proof: Let i^* be some facility opened by S^*. Then consider adding i^* to S, and connecting all clients of i^* in S^* to i^*. Since S is a local optimum, the addition of i^* cannot imporve the cost, therefore

$$\begin{aligned} f_{i^*} - \sum_{j:\sigma^*(j)=i^*} (r^*(j) - r(j)) > 0 \\ \sum_{i^* \in S^*} \left[f_{i^*} - \sum_{j:\sigma^*(j)=i^*} (r^*(j) - r(j)) \right] > 0 \\ C_f(S^*) + C_r(S^*) - C_r(S) > 0 \\ C(S^*) - C_r(S) > 0 \end{aligned}$$

■

Now that a bound on $C_r(S)$ is established all thats left is to bound $C_f(S)$

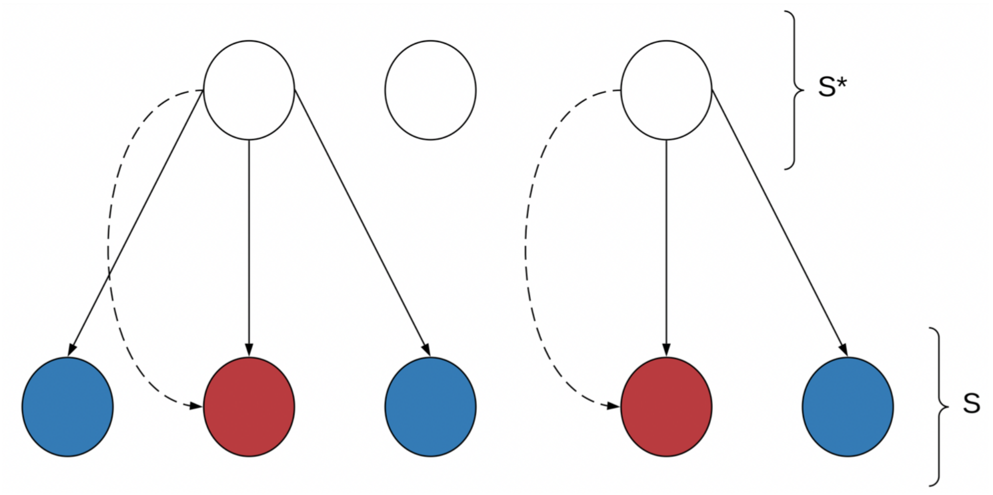

Figure 4.3.1: Assingment of vetices for lemma 4.3.5, red nodes are prinary, blue are secondary

Lemma 4.3.5 $C_f(S) \leq 2C(S^*) + 2C_r(S^*) \leq 4C(S^*)$

Proof: We begin by assining each vertex $i \in S$ to a vertex $i^* \in S^*$, where each i maps to the closest i^* as defined by $c_{i,i*}$. From here we consider two possible cases. Potentially, multiple vertices i in S can be assinged to a single i^*. We say that if some i is assinged to i^* and closest to i^* among all other facilities assinged to i^* it is **primary**, otherwise we call it **secondary**. We will provide separate arguments for the cost of connecting clients to particular i.

Claim 4.3.6 *If $i \in S$ is **primary**, then $f_i \leq R_i + R_i^* + f_{i^*}$*

Proof: We will proceed by swapping some facility i in S with i^*, the facility closest to i in S^*. Then, since S is a local optima, the swap should increase the cost of the solution. Since S is locally optimal, we compute the cost of swapping i and i^*

$$f_i^* - f_i + \sum_{j:\sigma(j)=i} c_{i^*,j} - c_{i,j} \geq 0$$

We not attempt to bound $c_{i^*,j} - c_{i,j}$ By the triangle inequality we obtain

$$ci^*, j - c_{i,j} \leq c_{i,i^*}$$

Considering that i^* is the closest facility to i in S^*

$$\begin{aligned} c_{i,i^*} &\leq c_{i,\sigma^*(j)} \\ &\leq c_{j,\sigma^*(j)} + cj, i \qquad \text{by triangle inequality} \\ &= r_i + r_i^* \qquad \text{by definition} \end{aligned}$$

Having obtained the upper bound we return to the original equasion

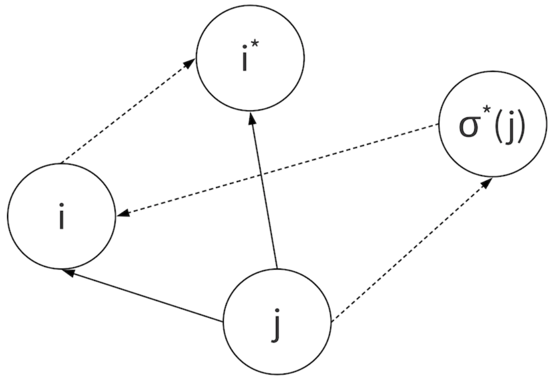

Figure 4.3.2: Supporting diagram for claim 4.3.6

$$
\begin{aligned}
f_i^* - f_i + \sum_{j:\sigma(j)=i} c_{i^*,j} - c_{i,j} \geq 0 \\
f_i^* - f_i + \sum_{j:\sigma(j)=i} r_i + r_i^* \geq 0 \\
f_i^* - f_i + R_i + R_i^* \geq 0 \\
f_i \leq R_i + R_i^* + f_{i^*}
\end{aligned}
$$

This completes our proof of the claim. ■

We now consider the case of i as a **secondary** vertex.

Claim 4.3.7 *If i is secondary, then $f_i \leq 2(R_i + R_i^*)$.*

Proof: Let i' denote the **primary** facility. In this case, consider removing i from S, and assinging customers of i to i'. In this case the local optimality of S still implies that the cost would not decrease, thus we calculate the cost of this change.

$$-f_i + \sum_{j:\sigma(j)=i} c_{i^*,j} - c_{i,j}$$

Similarly we attempt to upper bound $c_{i^*,j} - c_{i,j}$.

$$
\begin{aligned}
c_{i',j} - c_{i,j} &\leq c_{i,i^*} + c_{i^*,i'} && \text{by the triangle inequality} \\
&\leq 2c_{i,i^*} && \text{since i is secondary} \\
&\leq 2c_{i,\sigma^*(j)} && \text{by triangle inequality} \\
&= 2(r_i + r_i^*) && \text{by definition}
\end{aligned}
$$

Similarly to the proof above we return to the original claim

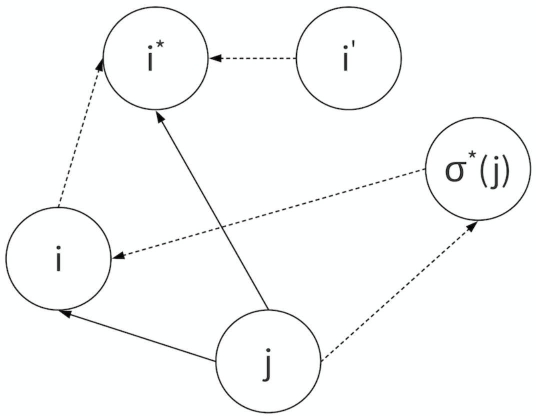

Figure 4.3.3: Supporting diagram for claim 4.3.7

$$
\begin{aligned}
-f_i + \sum_{j:\sigma(j)=i} c_{i^*,j} - c_{i,j} &\geq 0 \\
-f_i + \sum_{j:\sigma(j)=i} 2(r_i + r_i^*) &\geq 0 \\
-f_i + 2(R_i + R_i^*) &\geq 0 \\
f_i &\leq 2(R_i + R_i^*)
\end{aligned}
$$

■

Now finally we put the two results of the two claims above together

$$
\begin{aligned}
C_f(S) &\leq C_f(S^*) + \sum_{i \in S} 2(R_i + R_i^*) \\
&= C_f(S^*) + 2C_r(S) + 2C_r(S^*) \\
&\leq 2C(S^*) + 2C_r(S) \\
&\leq 4C(S^*) \qquad \text{by lemma 4.3.4}
\end{aligned}
$$

Thus the lemma is proven. ■

Finally we can put all the pieces together and prove theorem 4.3.3

$$C(S) = C_f(S) + C_r(S) \leq 4C(S^*) + C(S^*) = 5C(S^*)$$