CS787: Advanced Algorithms

Scribe: Yiying Zhang and Baris Aydinlioglu **Lecturer:** Shuchi Chawla
Topic: Local Search Based Appoximation **Date:** October 03, 2007

Today we talk about an important technique of approximation algorithms: local search. We first give fundamental ideas of local search approximation. Then we look into two problems where local search based approximation algorithms apply, max-cut problem and facility location problem.

10.1 Fundamentals of Local Search

The basic idea of local search is to find a good local optima quickly enough to approximate the global optima. This idea is somewhat like hill climbing. Starting from an arbitrary point, we try to go down or climb up a little step and compare the objective value at the new point to that at the old point. We then choose the step that increase the objective value. In this manner, we finally reach a local maximal. Such algorithm is also known as *ascending or descending algorithm.*

We can also explain local search method using a graph. We consider a graph, whose nodes are feasible solutions and whose edges connect two "neighboring" feasible solutions. Two feasible solutions are neighbors if from either one we can obtain the other by a single "local" step. What constitutes a local step depends on the problem, as we will see through examples later. Note that we do not construct the entire graph, as its size would usually be exponential.

Once the structure of the graph is decided, in order to find a local optimum, we start with an arbitrary node in the graph of feasible solutions, and we look at its neighbors to see if there is a node with a better objective value. If there is, we find we move to that node feasible solution in the graph. We don't necessarily look for the neighboring node with the best objective value; in most cases a better value is enough. We repeat this process until there is no neighbor with a better objective value. At that point we find the local optimum. See figure below.

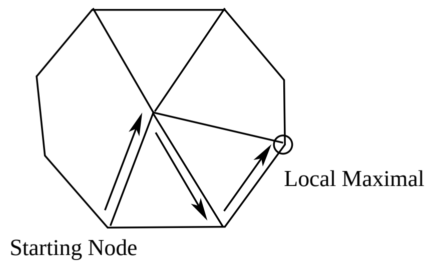

Figure 10.1.1: Graph Representation of Local Search

Now, the problem is how to design a good neighborhood structure for the graph. On one hand, we cannot allow for any node to be neighbor to too many other nodes. Otherwise in the extreme case we would get a complete graph and it would take exponential time to consider the neighbors of even a single node. On the other hand, we cannot have too few neighbors for a node, because we want any locally optimal node in the graph to have objective value within a factor of that of the global optimum.

In summary, we seek the following properties in a graph of feasible solutions:

1. Neighborhood of every solution is small
2. Local optima are nearly as good as global optima
3. We can find some local optima in a polynomial number of steps

The third point is often proven by showing that at each step we improve the objective value by some large enough factor. If the original solution has objective value V and the local optimal solution has objective value V^*, and at each step the objective value is improved by $(1+\epsilon)$, then we will have $\log_{(1+\epsilon)} \frac{V^*}{V}$ steps. We will give more details in the following examples.

10.2 Max-Cut Problem

Problem Definition:

Given: Unweighted graph $G = (V, E)$

Goal: Determine a cut $(S, \overline{S})$, where $S \neq V$ and $S \neq \emptyset$, such that $val(s) = \sum_{e \in E \cap (S \times \overline{S})} w_e = |E \cap (S \times \overline{S})|$ is maximized.

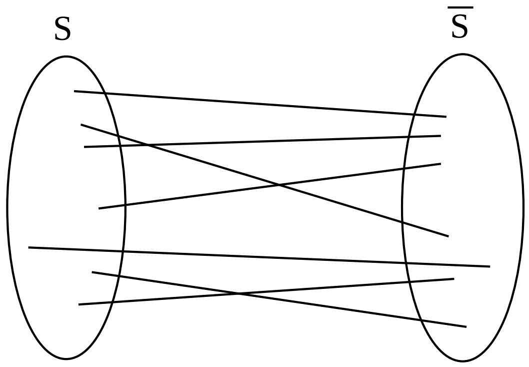

Figure 10.2.2: A cut (S, $\overline{S}$)

This problem is known to be NP-hard, which may appear surprising since its minimization version is efficiently solvable.

Since any nontrivial cut yields a feasible solution, we already know the nodes of our graph of feasible solutions. We need to determine a neighborhood structure on this graph, in other words we need to decide on what constitutes a local step.

Given a solution $(S, \overline{S})$, we consider a local step as moving a vertex from one side of the partition to another. So if v is in S (resp. $\overline{S}$), then moving v to $\overline{S}$ (S) would be a local step.

Our algorithm then is the following. Start with any nontrivial partition. Take local steps until a partition is reached such that none of its neighbors give a larger cut. Return this last partition.

Theorem 10.2.1 *The max-cut algorithm described above runs in $O(n^2)$ steps, where n is the number of vertices in the graph.*

Proof: At every step, we improve the size of the cut by at least 1. Therefore, the number of steps cannot be more than the total number of edges in the graph, which is $O(n^2)$. ■

Theorem 10.2.2 *If $(S, \overline{S})$ is a local optimal cut and $(S^*, \overline{S^*})$ is the global optimal cut, then $val(S) \geq \frac{1}{2} val(S^*)$.*

Proof: Let $(S, \overline{S})$ be locally optimal. For any vertex v in the graph, denote by $C(v)$ the number of neighbors of v that are across the cut. So if v is in S, then $C(v) = |(u, v) \in E : u \in \overline{S}|$. Similarly, denote by $C'(v)$ the number of neighbors of v that are in the same side of the cut as v. So if $v \in S$, then $C'(v) = |(u, v) \in E : u \in S|$.

Observe that for any vertex v in the graph we must have $C'(v) \leq C(v)$, for otherwise we can move v to the other partition and obtain a bigger cut, contradicting the local optimality of $(S, \overline{S})$.

Now, since $val(S)$ is the number of edges in the cut, the number of edges that are not in the cut is $|E| - val(S)$. Thus we have

$$val(S) = \frac{1}{2} \sum_{v \in V} C(v) \quad \geq \quad \frac{1}{2} \sum_{v \in V} C'(v) = |E| - val(S) \ .$$

This tells us that the number of edges in the cut is at least half of the total number of edges in the graph, $|E|$. But we know that $|E|$ is an upper bound on $val(S^*)$, the number of edges in the global max-cut. Therefore we have proved the claim. ■

Theorem 10.2.2 shows that our local search algorithm is a 2-approximation. There is another algorithm that yields a 2-approximation, which uses randomization. In that algorithm, we put a vertex in S or $\overline{S}$ at random (with a probability of $\frac{1}{2}$, v gets assigned to S). Then any edge in the graph has a probability of $\frac{1}{2}$ to be in the cut. So in expectation we get at least half of the edges in the cut. We will not go into details of this algorithm.

For a long while a less-than-2-approximation to this problem was not known, but now there is an algorithm that achieves a 1.3-approximation that uses sophisticated techniques. We will study this algorithm in a few weeks.

10.3 Facility Location

Most of the difficulty in the study of local search approximation algorithms tends to be in the analysis activity rather than in the design activity. We now move on to a problem that is more representative of this fact: a simple algorithm, but an involved analysis.

10.3.1 Problem Definition

Consider the following prominent problem in Operations Research, named **Facility Location**. A company wants to distribute its product to various cities. There is a set I of potential locations where a storage facility can be opened and a fixed "facility cost" f_i associated with opening a storage facility at location $i \in I$. (Sometimes we will say "facility i" to mean "facility at location i"). There also is a set J of cities to which the product needs to be sent, and a "routing cost" $c(i, j)$ associated with transporting the goods from a facility at location i to city j. The goal is to pick a subset S of I so that the total cost of opening the facilities and transporting the goods is minimized.

In short, we want to minimize

$$C(S) = C_f(S) + C_r(S) \ , \tag{10.3.1}$$

where

$$C_f(S) = \sum_{i \in S} f_i \ , \tag{10.3.2}$$

and

$$C_r(S) = \sum_{j \in J} \min_{i \in S} c(i, j) \ . \tag{10.3.3}$$

Notice that once a subset S of I is chosen, which facilities service which cities is automatically determined by the way $C_r(S)$ is defined.

In the way we have stated it so far, this problem is a generalized version of `SET COVER`, which can be seen by the following reduction: Given a `SET COVER` instance $(S, \mathcal{P})$, where $\mathcal{P} \subset 2^S$, map each $P \in \mathcal{P}$ to a facility and each $s \in S$ to a city, then set f_P to 1 and set $c(P, s)$ to either 0 or $+\infty$ depending on if $s \in P$ or not, respectively.

This reduction implies that the Facility Location problem would not allow a better approximation than what we can best do with `SET COVER`. In today's lecture we want to study a constant factor approximation to Facility Location, and for this purpose we impose some restrictions on the problem.

We restrict the costs $c(i, j)$ to form a *metric*. A metric is a function that is symmetric and that satisfies the triangle equality. For the costs in our problem this means (respectively) that we have $c(i, j) = c(j, i)$ for each $i, j \in I \cup J$, and that we have $c(i, j) \leq c(i, k) + c(k, j)$ for all $i, j, k \in I \cup J$. Notice that we have extended the notion of cost to in-between facilities and also cities. To make sense out of this, just think of costs as physical distances, and interpret equation (10.3.3) as each city being serviced by the nearest open facility.

10.3.2 A local search algorithm

Our local search algorithm works as follows. We pick any subset S of the set I of facilities. This gives us a feasible solution right away. We then search for the local neighborhood of the current feasible solution to see if there is a solution with a smaller objective value; if we find one we update our feasible solution to that one. We repeat this last step until we do not find a local neighbor that yields a reduction in the objective value. There are two types of "local steps" that we take in searching for a neighbor: i) remove/add a facility from/to the current feasible solution, or ii) swap a facility in our current solution with one that is not.

It is possible to consider variations on the local steps such as removing/adding more than one facility, or swapping d facilities out and bringing e facilities in, etc.. With more refined local steps, we can reach better local optima, but then we incur additional runtime due to the neighborhood graph getting denser. For our purposes the two local steps we listed above suffice, and indeed we show below that our algorithm with these two local steps give us a 5-factor approximation[1].

10.3.3 Analysis

Theorem 10.3.1 *Let S be a local optimum that the above algorithm terminates with, and S^* be a global optimum. Then*

$$C(S) \leq 5 \cdot C(S^*) \ .$$

We break up the proof of this theorem into two parts. First we show an upper bound on the routing cost of a local optimum, then we do the same for the facility cost.

Lemma 10.3.2

$$C_r(S) \leq C(S^*) \ .$$

Proof:

Consider adding some facility i^* in the globally optimal solution S^* to the locally optimal S to obtain S'. Clearly, $C(S) \geq C(S')$ because our algorithm must have explored S' as a local step before concluding that S is locally optimal. If i^* was already in S then there is no change. So we have

$$C_f(S') - C_f(S) + C_r(S') - C_r(S) \geq 0 \ .$$

Since $S' = S \cup \{i^*\}$, we rewrite the above as

$$f_{i^*} + C_r(S') - C_r(S) \geq 0 \ . \tag{10.3.4}$$

Before moving on we introduce some notation for brevity. Given (a feasible solution) $S \subset I$, a facility location i, and a city j, we define the following:

— $N_S(i)$ = the set of cities that are serviced by facility i in S,[2]

[1]Actually, these two steps give us a 3-factor approximation, but we do not cover that analysis here.

[2]For reasons that will become clear later, a better way to think of the "facility i services city j in S" relation is "facility i is the closest to j among all facilities in S". For brevity we will continue to speak of the "service" relation, but the latter interpretation lends itself better to generalizations and should be kept in mind.

— $\sigma^S(j)$ = the facility that services city j in S – i.e., facility i such that $j \in N_S(i)$,

— $\sigma^*(j) = \sigma^{S^*}(j)$,

— $\sigma(j) = \sigma^{S'}(j)$.

As mentioned earlier, by the way the objective function C was defined (see 10.3.3), given the set of facilities S', the decision of which city gets served by which facility is automatically determined: each city gets served by the nearest facility and indeed this is the best we can do with S'. We exploit this fact next, by doing worse with S'.

Consider $N_{S^*}(i^*)$, the set of cities that are serviced by i^* in the globally optimal solution S^*. Now, some of these cities may be serviced, in the solution S', by facilities other than i^*, because S' may contain facilities that are closer to them. What if we "re-route" S' so that each of these cities in $N_{S^*}(i^*)$ actually gets served by i^*, and not necessarily by the nearest open facility to it? Clearly we would incur a non-negative increase in the total routing cost of S'.

We can state the last argument as it corresponds to the problem formulation. Consider $C_r(S')$, the routing costs of S'. $C_r(S')$ is the sum of the distances of each city to a nearest facility in S'. What we really mean by "re-route" is to just take some terms in this sum and substitute it with terms that are not smaller than the ones we take out. In particular, for each city $j \in N_{S^*}(i^*)$, we take out $c(\sigma(j), j)$, the shortest distance between j and a facility in S', and instead substitute it with $c(i^*, j)$, a possibly larger–but surely not smaller–distance. Therefore the resulting sum is not smaller than $C_r(S')$. Call this new sum $\bar{C}_r(S')$.

Observe that by construction, $\bar{C}_r(S')$ differs from $C_r(S)$ only in the terms $j \in N_{S^*}(i^*)$, and that we have:

$$\bar{C}_r(S') - \sum_{j \in N_{S^*}(i^*)} c(i^*, j) \;=\; C_r(S) - \sum_{j \in N_{S^*}(i^*)} c(\sigma(j), j) \ .$$

Combining the above with (10.3.4), we obtain:

$$0 \;\leq\; f_{i^*} + \bar{C}_r(S') - C_r(S) \;=\; f_{i^*} + \sum_{j \in N_{S^*}(i^*)} [c(i^*, j) - c(\sigma(j), j)] \ .$$

Repeating this for all $i^* \in S^*$ and summing up, we get

$$\sum_{i^* \in S^*} \left[f_{i^*} + \sum_{j \in N_{S^*}(i^*)} [c(i^*, j) - c(\sigma(j), j)] \right] \geq 0 \ .$$

Rearranging, we get

$$\sum_{i^* \in S^*} f_{i^*} + \sum_{\substack{i^* \in S^* \\ j \in N_{S^*}(i^*)}} [c(i^*, j) - c(\sigma(j), j)] \geq 0 \ .$$

Now, observing that the double sum in the last inequality is going through all the cities, we replace it with a single sum and write the inequality as

$$\sum_{i^* \in S^*} f_{i^*} + \sum_{j \in J} [c(\sigma^*(j), j) - c(\sigma(j), j)] \geq 0 \ ,$$

which is the same thing as

$$C_f(S^*) + C_r(S^*) - C_r(S) \geq 0 \ ,$$

proving the lemma. ■

Lemma 10.3.3

$$C_f(S) \leq 2 \cdot C(S^*) + 2 \cdot C_r(S) \leq 4 \cdot C(S^*) \ .$$

Proof: We begin with some more notation. In the proof of the previous lemma we defined $\sigma(j)$ to be a function of cities. Now we extend it to facility locations. Given a solution S, a facility location i, we define:

– $\sigma^S(i) = \mathrm{argmin}_{i' \in S}\, c(i, i')$. In words, the nearest facility in S to i.

– $\sigma^*(i) = \sigma^{S^*}(i)$.

The rest of the proof proceeds somewhat along the lines of the proof of the previous lemma. Therefore we omit the details when we employ the same type of arguments as before.

Consider swapping some facility i in the locally optimal solution S with $\sigma^*(i)$, the nearest facility in S^* to i. Clearly, this gives us a solution S' with an objective function value larger than or equal to that of S, because S is locally optimal. Note that if $\sigma^*(i)$ was already in S, there is no change.

With i swapped out and $\sigma^*(i)$ swapped in, we now consider the following. If we "route" the goods in S' so that each city that was in $N_S(i)$ now gets served by $\sigma^*(i)$, and not necessarily by the nearest open facility to it, then this would again yield a non-negative change in the objective function value of S'. As a result, the combined increase in the cost of this configuration, call it δ, would be

$$0 \leq \delta = f_{\sigma^*(i)} - f_i + \sum_{j \in N_s(i)} [c(j, \sigma^*(i)) - c(j, i)] \ .$$

Recalling that the function c is a metric, we observe the following inequality regarding the term inside the summation in the last equation:

$$c(j, \sigma^*(i)) - c(j, i) \quad \leq \quad c(i, \sigma^*(i)) \quad \leq \quad c(i, \sigma^*(j)) \quad \leq \quad c(i, j) + c(j, \sigma^*(j))$$

The first and last inequalities follow from the triangle inequality. The second inequality follows from the fact that in S^*, the distance between city i and its serving facility $\sigma^*(i)$ is no larger than that between i any other facility in S^*, in particular $\sigma^*(j)$. (Here we have used the σ^* function with both a city and a facility location as argument). We now have

$$0 \leq \delta \leq f_{\sigma^*(i)} - f_i + \sum_{j \in N_S(i)} [c(j, i) + c(j, \sigma^*(j))] \ .$$

Expanding the summation on the right, we first get $\sum_{j \in N_S(i)} c(j, i)$, which is the routing cost of the cities served by i in S; denote this by R_i. Second, we get $\sum_{j \in N_S(i)} c(j, \sigma^*(j))$, which is the routing cost of the cities served by i in S^*; denote this by R_i^*.

Now we can rewrite the last inequality as

$$0 \leq \delta \leq f_{\sigma^*(i)} - f_i + R_i + R_i^*$$

Summing this equation over all $i \in S$, we obtain

$$\sum_{i \in S} f_{\sigma^*(i)} - C_f(S) + C_r(S) + C_r(S^*) \geq 0 \ .$$

We are not quite done yet. The first sum is counting some facilities in S^* more than once and not counting some at all. We will finish this proof in the next lecture.

■