Local Search

Anil Maheshwari

anil@scs.carleton.ca
School of Computer Science
Carleton University
Canada

Outline

Problem Statement

Techniques for Designing Approximation Algorithms

- Greedy
- Random Permutation
- Local Search
- Linear Programming Relaxation
- ...

Local Search - Technique for Designing Approximation Algorithms

An alternate to greedy algorithms for combinatorial optimization problems.

Approach:

- Find a feasible solution
- Keep swapping a constant number of objects from the current (local) solution to improve the objective function while maintaining the feasibility
- Stop when no more local improvements can be made
- Output the local solution

Analysis:

- Termination
- Quality of the resulting solution

Sample Problems:

1. Single Swaps:
   - 2-approximation algorithm for max cuts in graphs.
   - 5-approximation algorithm for metric k-median problem.
2. Multiple Swaps:
   - $(1+\epsilon)$-approximation algorithm for geometric hitting set.

Max-Cut

Max-Cut Problem:
Input: An undirected graph $G = (V, E)$.
Output: Find a subset $S \subset V$ such that the number of edges between S and $\bar{S} = V \setminus S$ is maximized. The subset S maximizing the number of edges between S and $\bar{S}$ is called the *Max-Cut* of G.

Weighted Max-Cut:
Input: An undirected graph $G = (V, E)$, where each edge has a positive integer weight.
Output: Find a subset $S \subset V$ such that the sum total of the weights on the edges between S and $\bar{S} = V \setminus S$ is maximized. The subset S maximizing the total weight of edges between S and $\bar{S}$ is called the *Weighted Max-Cut* of G.

Input: An undirected graph $G = (V, E)$.
Output: Find a subset $S \subset V$ such that the number of edges between S and $\bar{S} = V \setminus S$ is maximized. Let $cut(S, \bar{S})$ denote the number of edges between S and $\bar{S}$.

A Local Improvement Algorithm

1. Pick any vertex $v \in V$ and set $S \leftarrow \{v\}$ and $\bar{S} = V \setminus S$.
2. If $\exists v \in \bar{S}$ such that $cut(S \cup \{v\}, \bar{S} \setminus \{v\}) > cut(S, \bar{S})$, set $S \leftarrow S \cup \{v\}$ and $\bar{S} \leftarrow \bar{S} \setminus \{v\}$.
3. If $\exists v \in S$ such that $cut(S \setminus \{v\}, \bar{S} \cup \{v\}) > cut(S, \bar{S})$, set $S \leftarrow S \setminus \{v\}$ and $\bar{S} \leftarrow \bar{S} \cup \{v\}$.
4. Repeat Steps 2 and 3 until the size of the cut doesn't increases.
5. Report $S, \bar{S}, cut(S, \bar{S})$.

Analysis of the Local Improvement Algorithm

1. Pick any vertex $v \in V$ and set $S \leftarrow \{v\}$ and $\bar{S} = V \setminus S$.
2. If $\exists v \in \bar{S}$ such that
   $cut(S \cup \{v\}, \bar{S} \setminus \{v\}) > cut(S, \bar{S})$, set $S \leftarrow S \cup \{v\}$ and $\bar{S} \leftarrow \bar{S} \setminus \{v\}$.
3. If $\exists v \in S$ such that
   $cut(S \setminus \{v\}, \bar{S} \cup \{v\}) > cut(S, \bar{S})$, set $S \leftarrow S \setminus \{v\}$ and $\bar{S} \leftarrow \bar{S} \cup \{v\}$.
4. Repeat Steps 2 and 3 until the size of the cut doesn't increases.
5. Report $S, \bar{S}, cut(S, \bar{S})$.

Termination

The algorithm terminates in $O(|E|)$ steps.

Proof: In each iteration of Steps 2 or 3, the size of the cut increases by at least 1. Since, the max-cut size is at most $|E|$, the algorithm terminates in $O(|E|)$ iterations. $\square$

Analysis of the Local Improvement Algorithm

Size of Cut

The cut computed by the local improvement algorithm has $\geq \frac{|E|}{2}$ edges.

Proof: Let $(S, \bar{S})$ be the cut computed by the algorithm.
Consider any vertex $v \in S$. Let v has d_v neighbors. Using the local-optimality condition, at least $\frac{d_v}{2}$ neighbors of v are in $\bar{S}$ (otherwise, we can improve the solution).
The same argument applies for any vertex $v \in \bar{S}$. Thus,

$$\begin{aligned} cut(S, \bar{S}) &= \frac{1}{2} \sum_{v \in V} v's \text{ edges crossing the cut} \\ &\geq \frac{1}{2} \sum_{v \in V} \frac{d_v}{2} \\ &= \frac{1}{2} \frac{2|E|}{2} = \frac{|E|}{2} \qquad \square \end{aligned}$$

Local Improvement Algorithm

Theorem

The local improvement algorithm is a 2-approximation algorithm for the Max-Cut problem. The algorithm runs in polynomial time.

Consider the Weighted Max-Cut Problem

Input: An undirected graph $G = (V, E)$, where each edge has a positive integer weight.
Output: Find a subset $S \subset V$ such that the sum total of the weights on the edges between S and $\bar{S} = V \setminus S$ is maximized.

Question: Can we use the local improvement algorithm to find an approximation for the weighted max-cut?

Weighted Max-Cut Problem

For each edge $e \in E$, let w_e be its positive integer weight.
For a subset $S \subset V$, define the weight of $cut(S, \bar{S})$ as
$w(S, \bar{S}) = \sum_{e=uv \in E, u \in S, v \in \bar{S}} w_e$.

Local Improvement Algorithm (with weights):

1. Pick any vertex $v \in V$ and set $S \leftarrow \{v\}$ and $\bar{S} = V \setminus S$.
2. If $\exists v \in \bar{S}$ such that $w(S \cup \{v\}, \bar{S} \setminus \{v\}) > w(S, \bar{S})$, set $S \leftarrow S \cup \{v\}$ and $\bar{S} \leftarrow \bar{S} \setminus \{v\}$.
3. If $\exists v \in S$ such that $w(S \setminus \{v\}, \bar{S} \cup \{v\}) > w(S, \bar{S})$, set $S \leftarrow S \setminus \{v\}$ and $\bar{S} \leftarrow \bar{S} \cup \{v\}$.
4. Repeat Steps 2 and 3 until the weight of the cut stops increasing.
5. Report $S, \bar{S}, w(S, \bar{S})$.

Analysis

Let $W = \sum_{e \in E} w_e$.

- Termination: In each iteration, $w(S, \bar{S})$ increases by at least one unit, as all weights are integers.
  $\implies$ Algorithm terminates in at most $O(W)$ steps.
- Approximation factor - Is it true that for each vertex $v \in S$, $\sum_{e=uv \in E, u \in \bar{S}} w_e \geq \frac{1}{2} \sum_{e=vw \in E} w_e$?
- Is the running time polynomial in input parameters?
  What if we double the weight of an edge?

Modified Local Improvement Algorithm

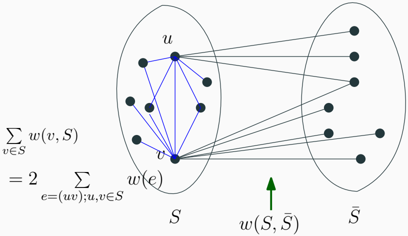
Let $\epsilon > 0$ be a parameter, and let $n = |V|$.

1. Pick the vertex $v \in V$ that has the maximum sum total of the weights of edges incident to it. Set $S \leftarrow \{v\}$ and $\bar{S} = V \setminus S$.
2. If $\exists v \in \bar{S}$ such that $w(S \cup \{v\}, \bar{S} \setminus \{v\}) \geq (1 + \frac{\epsilon}{n})w(S, \bar{S})$, set $S \leftarrow S \cup \{v\}$ and $\bar{S} \leftarrow \bar{S} \setminus \{v\}$.
3. If $\exists v \in S$ such that $w(S \setminus \{v\}, \bar{S} \cup \{v\}) \geq (1 + \frac{\epsilon}{n})w(S, \bar{S})$, set $S \leftarrow S \setminus \{v\}$ and $\bar{S} \leftarrow \bar{S} \cup \{v\}$.
4. Repeat Steps 2 and 3 until the weight of the cut stops increasing.
5. Report $S, \bar{S}$ and $w(S, \bar{S})$.

Analysis of Modified Local Improvement Algorithm

We make following observations:

1. Let $(S, \bar{S})$ be the cut returned by the algorithm.
2. For each vertex $v \in S$, by local optimality, we have
   $w(S, \bar{S}) \geq w(S \setminus \{v\}, \bar{S} \cup \{v\}) - \frac{\epsilon}{n} w(S, \bar{S})$
   $\implies w(v, \bar{S}) \geq w(v, S) - \frac{\epsilon}{n} w(S, \bar{S}) \qquad (*)$
3. Similarly, for each vertex $v \in \bar{S}$, we have
   $w(S, \bar{S}) \geq w(S \cup \{v\}, \bar{S} \setminus \{v\}) - \frac{\epsilon}{n} w(S, \bar{S})$
   $\implies w(v, S) \geq w(v, \bar{S}) - \frac{\epsilon}{n} w(S, \bar{S}) \qquad (**)$
4. Compute the sum total of all the inequalities (*) over all the vertices in S:
   $w(S, \bar{S}) \geq \sum_{v \in S} w(v, S) - |S| \frac{\epsilon}{n} w(S, \bar{S}) = 2 \sum_{e=uv; u,v \in S} w(e) - |S| \frac{\epsilon}{n} w(S, \bar{S})$



Analysis of Modified Local Improvement Algorithm (contd.)

- Similarly, the sum total of all the inequalities (**) for all the vertices in $\bar{S}$
$w(S,\bar{S}) \geq 2 \sum_{e=uv;u,v\in\bar{S}} w(e) - |\bar{S}|\frac{\epsilon}{n}w(S,\bar{S})$

- Adding the last two inequalities we obtain
$2w(S,\bar{S}) \geq 2\sum_{e=uv;u,v\in S} w(e) + 2\sum_{e=uv,u,v\in\bar{S}} w(e) - |S|\frac{\epsilon}{n}w(S,\bar{S}) - |\bar{S}|\frac{\epsilon}{n}w(S,\bar{S})$
Simplifying,

$$\begin{aligned} w(S,\bar{S}) &\geq \sum_{e=uv;u,v\in S} w(e) + \sum_{e=uv;u,v\in\bar{S}} w(e) - \frac{\epsilon}{2}w(S,\bar{S}) \\ &= (W - w(S,\bar{S})) - \frac{\epsilon}{2}w(S,\bar{S}) \end{aligned}$$

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

Analysis of Modified Local Improvement Algorithm (contd.)

Weight of any cut is upper bounded by W, including the weight of an optimal cut. Thus, we have

Claim

The modified local improvement algorithm is $\frac{1}{2+\epsilon}$ approximation algorithm for the weighted max-cut problem.

Next we analyze the running time.

Analysis of Modified Local Improvement Algorithm (contd.)

- Assume that the algorithm runs for k iterations and the sets computed by the algorithm are $S_0, S_1, S_2, \ldots, S_k$.
- Observe that $w(S_i, \bar{S}_i) \geq (1 + \frac{\epsilon}{n}) w(S_{i-1}, \bar{S}_{i-1})$, for $i = 1, \ldots, k$.
  $\implies w(S_k, \bar{S}_k) \geq (1 + \frac{\epsilon}{n})^k w(S_0, \bar{S}_0)$
- We know that $w(S_0, \bar{S}_0) \geq \frac{W}{n}$ and $W(S_k, \bar{S}_k) \leq W$.
- Thus, $W \geq W(S_k, \bar{S}_k) \geq (1 + \frac{\epsilon}{n})^k w(S_0, \bar{S}_0) \geq (1 + \frac{\epsilon}{n})^k \frac{W}{n}$
  $\implies k \leq \frac{\log n}{\log(1+\frac{\epsilon}{n})}$

If $\frac{\epsilon}{n} < 1$, $\log(1 + \frac{\epsilon}{n}) \geq \frac{\epsilon}{2n}$ (i.e., $\log(1 + x) > x/2$ for small values of x).

Thus, $k \leq \frac{\log n}{\log(1+\frac{\epsilon}{n})} \leq 2\frac{n}{\epsilon} \log n$.

Theorem

A $\frac{1}{2+\epsilon}$ approximation of maximum weight cut can be computed in polynomial time. The running time depends on $\frac{1}{\epsilon}$, $|V|$, and $|E|$.

k-Median

k-median Problem

Let $G = (V, E)$ be a complete graph on n vertices, where the costs on edges ($d : V \times V \to \Re^+$) satisfy the metric properties:

- $\forall u \in V : d(u, u) = 0$
- $\forall u, v \in V : d(u, v) = d(v, u)$
- $\forall u, v, w \in V : d(u, v) \leq d(u, w) + d(w, v)$

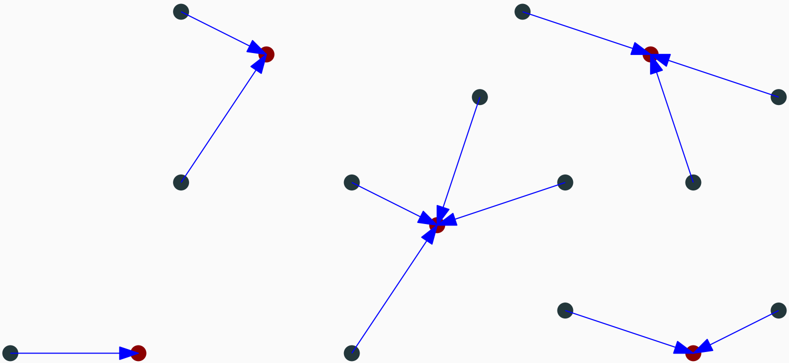
Definitions:

1. Facilities: Let $F \subset V$ such that $|F| = k$.
2. Distance to nearest facility: $d(v, F) = \min_{f \in F} d(v, f)$.
3. $cost(F) = \sum_{v \in V} d(v, F)$

k-median problem

Given the metric complete graph $G = (V, E)$, find $F \subset V$, $|F| = k$, such that $cost(F)$ is minimum.

An Example: Euclidean distance among points in plane, $k = 5$



Local Search Algorithm for k-median problem

Input: A metric graph $G = (V, E)$ and an integer $< 0 < k \leq |V|$
Output: $F \subset V$ such that $|F| = k$.

Step 1 (Initialize) $F \leftarrow \emptyset$. Select any k vertices from V. Add them to F as the initial set of k facilities.

Setp 2 (Local improvement step)
While there exists a pair of vertices (u, v), where $u \in V \setminus F$ and $v \in F$, such that $cost(F \setminus \{v\} \cup \{u\}) < cost(F)$,
$F \leftarrow F \setminus \{v\} \cup \{u\}$.

Step 3 Report F.

Approximation Quality

Let F^* be an optimal set of k-facilities for the k-median problem on the metric graph G. The set F returned by the local search algorithm satisfies $cost(F) \leq 5cost(F^*)$, i.e., it results in a 5-approximation.

Swap Pairs

- In Step 2 of the algorithm, if we make a swap (u, v), then the $cost(F)$ improves, i.e., $cost(F \setminus \{v\} \cup \{u\}) < cost(F)$.
- After the algorithm terminates, there doesn't exist any improving swap pairs. I.e., for any pair of vertices (u, v), where $u \in V \setminus F$ and $v \in F$, $cost(F \setminus \{v\} \cup \{u\}) \geq cost(F)$.
- To show $cost(F) \leq 5cost(F^*)$, we will select a set of specific non-improving swap pairs using the vertices in an optimal solution F^* and the solution F returned by the algorithm.

Finding a Select Set of Non-improving Swap Pairs

Let $F^* = (f_1^*, \ldots, f_k^*) \subset V$ be an optimal solution.
Let $F = (f_1, \ldots, f_k) \subset V$ is the solution returned by the algorithm.

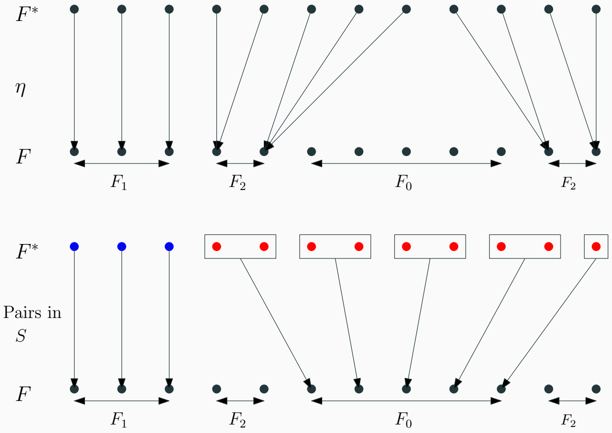
Define a mapping $\eta : F^* \to F$, that maps each facility (vertex) in F^* to the nearest facility in F.

We partition $F = F_0 \cup F_1 \cup F_{\geq 2}$ based on the in-degree of function η:
$F_0 = \{f \in F|$ no facilities in F^* maps to $f\}$
$F_1 = \{f \in F|$ exactly one facility in F^* maps to $f\}$
$F_{\geq 2} = \{f \in F|$ at least two facilities in F^* maps to $f\}$

Define the set $S \subset F^* \times F$ consisting of the following non-improving pairs of facilities:

1. All pairs corresponding to F_1 are in S. I.e. for each pair (f^*, r), where $r \in F$ and $f^* \in F^*$ and $\eta^{-1}(r) = f^*$, $(f^*, r) \in S$.
2. For the remaining facilities in F^*, pair them up, and assign each pair to a unique facility in F_0.

Illustration of Set S



An Observation on $|F_0|$

Are there enough facilities in F_0 so that the pairs of the remaining facilities in F^* can be assigned to the unique facilities in F_0?

Size of F_0

$|F_0| \geq \frac{|F|-|F_1|}{2}$

Proof: Use the following remarks to arrive at a proof

- $k = |F| = |F^*|$
- $|F| = |F_0| + |F_1| + |F_{\geq 2}|$
- The number of remaining facilities in F^* are $k - |F_1| = |F| - |F_1|$.
- Nearest neighbors of the remaining facilities in F^* are among $F_{\geq 2}$.
- Each facility in $F_{\geq 2}$ is near neighbor of at least two facilities of F^*.

$\square$

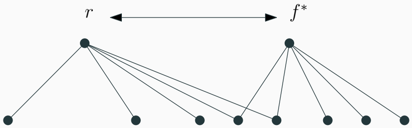
Bounding $cost(F)$ - Useful Notations

- Functions $\phi : V \to F$ and $\phi^* : V \to F^*$ maps vertices to the nearest facilities in F and F^*, respectively. If $\phi(v) = r$, than the nearest vertex of v in F is r.
- For any vertex $v \in V$, define the cost to the nearest facility in F^* by $O_v = d(v, F^*) = d(v, \phi^*(v))$. Similarly, define $A_v = d(v, F) = d(v, \phi(v))$.
- $cost(F^*) = \sum\limits_{v \in V} O_v$ and $cost(F) = \sum\limits_{v \in V} A_v$
- Define neighborhoods of facilities as the vertices that they serve. For each facility $f^* \in F^*$, we have $N^*(f^*) = \{v \in V | \phi^*(v) = f^*\}$. Similarly, for $r \in F$, $N(r) = \{v \in V | \phi(r) = f\}$.
- If $F^* = (f_1^*, \ldots, f_k^*)$, then $N^*(f_1^*), \ldots, N^*(f_k^*)$ is a partition of V. Similarly, $N(r_1), \ldots, N(r_k)$ is partition of V with respect to facilities in $F = \{r_1, \ldots, r_k\}$.

Main Claim

Consider a (non-improving) swap pair $(f^*, r) \in S$. Suppose we bring in the facility $f^* \in F^*$ and remove r from F, i.e., $F = F \cup \{f^*\} \setminus \{r\}$. The cost of the resulting k-median solution satisfies

$$\sum_{v \in N^*(f^*)} (O_v - A_v) + \sum_{v \in N(r)} 2O_v \geq cost(F \cup \{f^*\} \setminus \{r\}) - cost(F) \quad (1)$$



Proof comes later.

First we show that by summing Equation 1 over all the swap pairs in S, we have $cost(F) \leq 5cost(F^*)$ as follows.

5-approximation Bound From the Main Claim

Theorem

Suppose for each swap pair $(f^*, r) \in S$ we have
$\sum_{v \in N^*(f^*)} (O_v - A_v) + \sum_{v \in N(r)} 2O_v \geq cost(F \cup \{f^*\} \setminus \{r\}) - cost(F)$, than
$cost(F) \leq 5cost(F^*)$.

Proof:

1. Each $f^* \in F^*$ appears exactly once in S, and $\bigcup_{f^* \in F^*} N^*(f^*)$ partitions $V \implies \sum_{(f^*,r) \in S} \sum_{v \in N^*(f^*)} (O_v - A_v) \leq cost(F^*) - cost(F)$.
2. Each $r \in F$ appears at most twice in S. Thus, $\sum_{(f^*,r) \in S} \sum_{v \in N(r)} O_v \leq 2cost(F^*)$.
3. Since each pair in S is non-improving we have $cost(F \cup \{f^*\} \setminus \{r\}) - cost(F) \geq 0$
4. Summing for all pairs $(f^*, r) \in S$ the inequality $\sum_{v \in N^*(f^*)} (O_v - A_v) + \sum_{v \in N(r)} 2O_v \geq cost(F \cup \{f^*\} \setminus \{r\}) - cost(F)$, and apply 1-3, we obtain $cost(F^*) - cost(F) + 2 * 2cost(F^*) \geq 0$. Thus, $cost(F) \leq 5cost(F^*)$ $\square$

Proof of Main Claim

For a swap pair $(f^*, r) \in S$ we want to show
$$\sum_{v \in N^*(f^*)} (O_v - A_v) + \sum_{v \in N(r)} 2O_v \geq cost(F \cup \{f^*\} \setminus \{r\}) - cost(F).$$

Proof Sketch:

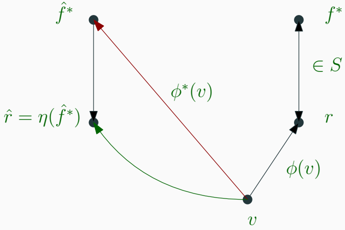
1. Note that we are swapping r by f^* in F. We are interested to upper bound $cost(F \cup \{f^*\} \setminus \{r\}) - cost(F)$.
2. We need to reassign facilities to some of the vertices because of this swap. For example, all vertices in $N(r)$ need to find a facility in $F \cup \{f^*\} \setminus \{r\}$.
3. We will assign each vertex in $N^*(f^*)$ to f^* in $F \cup \{f^*\} \setminus \{r\}$. The expression $\sum_{v \in N^*(f^*)} (O_v - A_v)$ accounts for the difference in the costs, as we save A_v from their costs but they costs us O_v.

Proof of Main Claim (contd.)

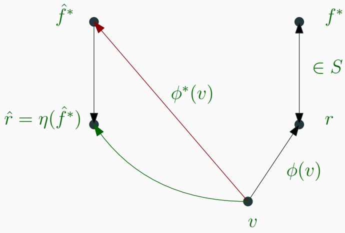
5. Note that there may be a vertex $v \in N^*(f^*)$ that ideally isn't served by f^* in $F \cup \{f^*\} \setminus \{r\}$. The reason is that $r' \in F \setminus \{r\}$ may be closer to v than f^*. Nevertheless we assign v to f^*, since we are trying to find an upper bound ($O(v) \geq d(v, r') \implies O_v - A_v \geq d(v, r') - A_v$).
6. All the vertices in $N(r) \cap N^*(f^*)$ are assigned to f^* in $F \cup \{f^*\} \setminus \{r\}$. By the similar upper bound argument, even if for a vertex $v \in N(r) \cap N^*(f^*)$ its nearest neighbor in $F \cup \{f^*\} \setminus \{r\}$ may not be f^*, but the same upper bound argument holds.
7. For each vertex $v \in N(r) \setminus N^*(f^*)$, assign it to its nearest neighbor in $F \cup \{f^*\} \setminus \{r\}$.
8. How to account for the costs of members in $N(r) \setminus N^*(f^*)$?
9. Let $v \in N(r) \setminus N^*(f^*)$.

Proof of Main Claim (contd.)

11. Since v isn't served by f^* in optimal $\implies$ v is served by a facility $\hat{f}^* \in F^*$, i.e., $\phi^*(v) = \hat{f}^*$.
12. Either $\hat{f}^* \in F$ or $\hat{f}^* \notin F$.
13. If $\hat{f}^* \in F$: then we assign v to $\hat{f}^*$.
14. If $\hat{f}^* \notin F$, consider $\hat{r} = \eta(\hat{f}^*)$, i.e. nearest neighbor of $\hat{f}^*$ in F. (Note: $\hat{r} \neq r$. If it is, than $r \in F_1 \cup F_{\geq 2}$, we wouldn't have assigned f^* to r.) Assign v to $\hat{r}$.



Proof of Main Claim (contd.)



By triangle inequality: $d(v, \hat{r}) \leq d(v, \hat{f}^*) + d(\hat{f}^*, \hat{r})$.
-Subtracting $d(v, r)$ from both the sides, we get
$d(v, \hat{r}) - d(v, r) \leq d(v, \hat{f}^*) + d(\hat{f}^*, \hat{r}) - d(v, r)$.
- We know that $d(\hat{f}^*, \hat{r}) \leq d(\hat{f}^*, r)$ because of nearest neighbor function η.
- Thus, $d(v, \hat{r}) - d(v, r) \leq d(v, \hat{f}^*) + d(\hat{f}^*, r) - d(v, r)$.
- By triangle inequality, $d(\hat{f}^*, r) - d(v, r) \leq d(v, \hat{f}^*)$.
- Thus, $d(v, \hat{r}) - d(v, r) \leq d(v, \hat{f}^*) + d(\hat{f}^*, r) - d(v, r) \leq 2d(v, \hat{f}^*) = 2O_v$.

$\square$

Running Time

Input: A metric graph $G = (V, E)$ and an integer $< 0 < k \leq |V|$
Output: $F \subset V$ such that $|F| = k$.

Step 1 (Initialize) $F \leftarrow \emptyset$. Select any k vertices from V and insert them in F.

Setp 2 (Local improvement step) While there exists a pair of vertices (u, v), where $u \in V \setminus F$ and $v \in F$, such that $cost(F \setminus \{v\} \cup \{u\}) < cost(F)$, set $F \leftarrow F \setminus \{v\} \cup \{u\}$.

Step 3 Report F.

Running Time:

1. In each execution of Step 2, the cost improves $\implies$ Algorithm terminates.
2. How many times Step 2 is executed?
3. Assume all $d(u, v)$ values are positive integers and let $\Delta = \sum\limits_{u,v} d(u, v)$.
4. Number of times Step 2 is executed $\leq \Delta$.
5. Modify Step 2: Swap if cost improves by at least a factor of $(1 - \frac{\epsilon}{poly(n)})$

Conclusions

Theorem

Let F^* be an optimal set of k-facilities for the k-median problem on the metric graph G. The set F returned by the local search algorithm satisfies $cost(F) \leq (5 + \epsilon)cost(F^*)$. Moreover, the algorithm runs in polynomial time. Run time depends on $|V|$ and $\frac{1}{\epsilon}$.

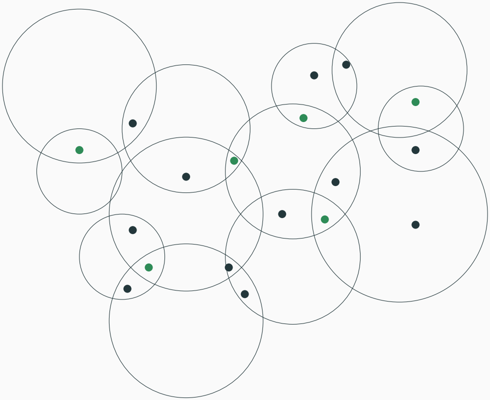
Improvements: In place of performing a single swap in Step 2, perform $t \geq 1$ multi-swaps. A refined analysis shows that $cost(F) \leq (3 + \frac{2}{t})cost(F^*)$.

Approximating Geometric Hitting Set

Geometric Hitting Set Problem

Input: A set $\mathcal{D}$ of disks and a set P of points in plane.
Output: Find a subset $S \subseteq P$ of smallest cardinality that hits all disks in D.

We say a point $p \in P$ *hits* the disk $D \in \mathcal{D}$ if $p \in D$.



k-level Local Search algorithm for finding a hitting set for disks:

Input: A set $\mathcal{D}$ of disks and a set P of points in plane. A (large) integer $k > 0$.
Output: A subset $S \subseteq P$ that hits all disks in $\mathcal{D}$.

1. Initialization: $S \leftarrow P$. Check if S hits all disks. If not, report infeasibility and stop.
2. Local Improvement Step: Keep replacing any set of k points in S by at most $k-1$ points of P so that points in S hits all disks in $\mathcal{D}$.
3. Return S.

Main Result

Let $S^* \subseteq P$ be an optimal hitting set for $\mathcal{D}$. The set S returned by the algorithm satisfies $|S| \leq (1 + \frac{c}{\sqrt{k}})|S^*|$, for some constant c.

Ingredients: Separators + Planar (Delaunay) Triangulations

Locality Condition

A bipartite graph for $B, R \subseteq P$

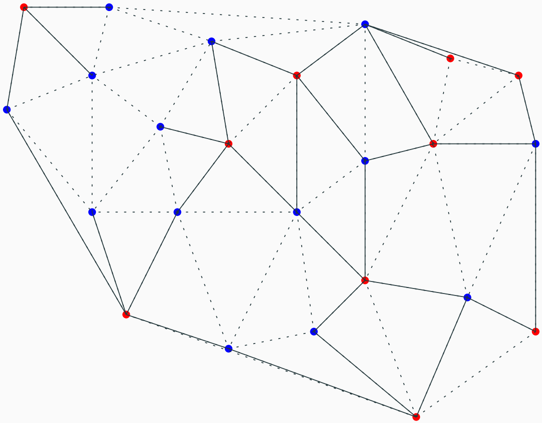
Let $B, R \subset P$ be subset of points of P, and let $G = (V = B \cup R, E)$ be a bipartite graph such that the following *locality condition* holds:
For any disk $D \in \mathcal{D}$, where $B \cap D \neq \emptyset$ and $R \cap D \neq \emptyset$, there exist points $b \in B \cap D$ and $r \in R \cap D$ such that $(b, r) \in E$.

Delaunay Triangulation on $B \cup R$

Let G be the planar graph corresponding to the Delaunay triangulation of $B \cup R$, where we only keep the edges between a pair of red and blue points. The graph G satisfies the locality condition.

Proof: By construction, G is bipartite.
If a disk $D \in \mathcal{D}$ contains points from both B and R, than there is a point $b \in B$ and $r \in R$ such that the Delaunay edge br completely lies inside D.
This uses the property that for a Delaunay triangulation, points within an arbitrary disk forms a connected subgraph. $\square$

Illustration of Delaunay Graph $G = (B \cup R, E)$



Neighborhoods in G

Neighborhoods

For each vertex $v \in G = (V, E)$, let $N(v)$ be all the vertices adjacent to v. For a subset of vertices $W \subset V$, define $N(W) = \bigcup_{v \in W} N(v)$.

Let $B = S$ be the set returned by the local search algorithm, and let $R = S^*$ be an optimal solution for the hitting set problem. Assume that $B \cap R = \emptyset$ (otherwise, we can remove the common points and the disks that they hit).

Note that B hits all disks in $\mathcal{D}$ and similarly R hits all disks in $\mathcal{D}$. Consider the planar bipartite graph $G = (B \cup R, E)$ formed using the Delaunay triangulation of $B \cup R$ and retain only the red-blue edges.

Neighborhoods in $G = (B \cup R, E)$

Claim 1

For any subset $B' \subset B$, $B \cup N(B') \setminus B'$ is a hitting set for $\mathcal{D}$.

Proof:
- Consider any disk $D \in \mathcal{D}$.
- Since points in B hits all disks, there is some point in B that hits D.
- If any of the points in $B \setminus B'$ hits $D \implies$ Points in $B \cup N(B') \setminus B'$ also hits D.
- Now, assume only the points in B' hits the disk D.
- Points in R also hits all disks in $\mathcal{D}$
- Let $r \in R$ hits D and let $b \in B'$ hits D.
- Both points $b, r \in D$.
- By the Delaunay property, there is a bichromatic edge in the Delaunay triangulation that completely lies in D.

$\implies$ The neighborhhod set of B' also includes a red point in R that is in the disk D.
- Thus, $B \cup N(B') \setminus B'$ is a hitting set for $\mathcal{D}$. $\square$

Claim 2 - Expansion Property

For every subset $B' \subseteq B$ of size $\leq k$ in the graph $G = (B \cup R, E)$, $|N(B')| \geq |B'|$, i.e. the size of the neighborhood of B' is at least $|B'|$.

Proof:

- The set B is obtained by executing the local search algorithm with parameter k

$\implies$ there doesn't exist any improving swaps, i.e. no set of k points (vertices) in B can be replaced by $k-1$ points from P to hit all the disks in $\mathcal{D}$.

- By Claim 1, the set $B \cup N(B') \setminus B'$ is a hitting set for $\mathcal{D}$.

$\implies$ $|N(B')| \geq |B'|$, otherwise the local optimality condition is violated.

$\square$

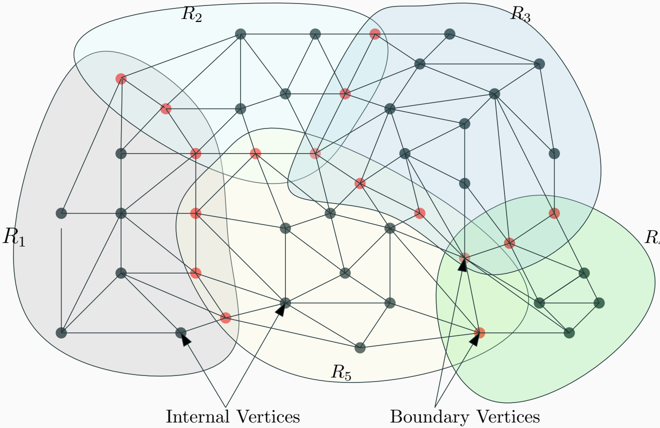
Fredrickson's r-partitioning of Planar Graphs

Let $G = (V, E)$ be a planar graph on n vertices, and let r be a number.
- Fredrickson, using the recursive application of Lipton and Tarjan's planar separator theorem, shows a division of planar graph in regions consisting of interior and boundary vertices.
- Each interior vertex is contained within a region and is adjacent to vertices within that region.
- Boundary vertices are shared between at least two regions.

Lemma

Let G be a planar graph on n vertices. A r-division divides G in $\Theta(n/r)$ regions, where each region consists of $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices. A r-division of a planar graph G can be computed in $O(n \log n)$ time.

Illustration of r-partitioning



Bounding the size of B

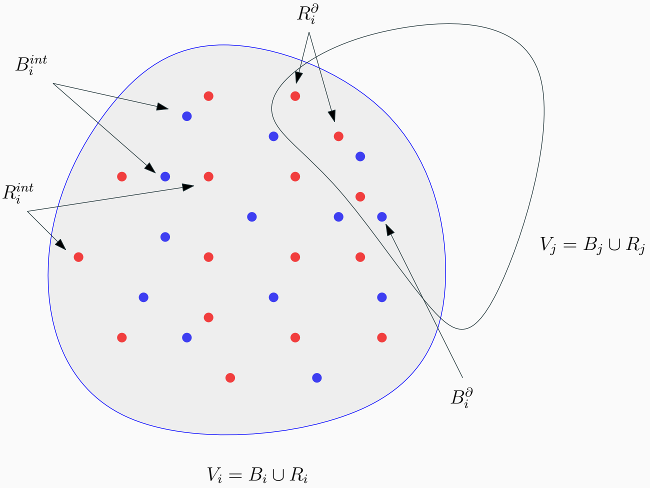
Main Claim

Let $S \subset P$ be the set of points returned by local search algorithm with parameter k and let $S^* \subset P$ be an optimal solution for the hitting set problem for the disks in $\mathcal{D}$ by points in P. We define the Delaunay triangulation on red-blue points where $B = S$ and $R = S^*$, and construct the bipartite graph $G = (B \cup R, E)$ by retaining only the edges between red and blue points. The following holds: $|B| \leq (1 + \frac{c}{\sqrt{k}})|R|$ for some constant c.

Proof:

- Assume $n = |B| + |R|$.
- We apply Fredrickson's r-partitioning on the graph G, where $r = k$.
- G is divided into $\Theta(n/k)$ regions, each region consisting of $\leq k$ vertices and $O(\sqrt{k})$ boundary vertices.
- The total number of boundary vertices is $O(n/\sqrt{k})$
- Let $V_i = B_i \cup R_i$ be the set of vertices in the i-th region in the partitioning.
- Let B_i^{int}, B_i^{∂} be the interior and boundary blue vertices in V_i
- Let R_i^{int}, R_i^{∂} be the interior and boundary red vertices in V_i

Illustration of Notation



Bounding the size of B (contd.)

- Sum total of boundary vertices among all regions is $\gamma n/\sqrt{k}$, where γ is a constant from Fredrickson's r-partitioning. .
- I.e., $\sum_i (|B_i^\partial| + |R_i^\partial|) \leq \gamma n/\sqrt{k}$.

- Number of interior blue vertices, $|B_i^{int}|$, in any region is at most k.
- By Claim 2 (Expansion Property), we know that $|B_i^{int}| \leq |N(B_i^{int})|$.

- What are the vertices in $N(B_i^{int})$?
- $N(B_i^{int}) \subseteq R_i^{int} \cup R_i^\partial$ - Thus we have $|B_i^{int}| \leq |R_i^{int}| + |R_i^\partial|$
- Add $|B_i^\partial|$ on both sides and we obtain: $|B_i^\partial| + |B_i^{int}| \leq |R_i^{int}| + |R_i^\partial| + |B_i^\partial|$

- Summing over all regions we have:

$$\sum_i \left(|B_i^\partial| + |B_i^{int}|\right) \leq \sum_i |R_i^{int}| + \sum_i \left(|R_i^\partial| + |B_i^\partial|\right) \tag{2}$$

- Note, $\sum_i \left(|B_i^\partial| + |B_i^{int}|\right) \geq |B|$, $|R| \geq \sum_i |R_i^{int}|$, and $\sum_i \left(|R_i^\partial| + |B_i^\partial|\right) = \gamma n/\sqrt{k} = \gamma(|B| + |R|)/\sqrt{k}$.

Bounding the size of B (contd.)

We have

$$|B| \leq \sum_i \left(|B_i^\partial| + |B_i^{int}|\right) \leq |R| + \gamma(|B| + |R|)/\sqrt{k} \tag{3}$$

Let $k \geq 4\gamma^2$ and $c = 4\gamma$. Note $\gamma/\sqrt{k} \leq 1/2$.

$$\begin{aligned}
|B| &\leq \left(\frac{1+\gamma/\sqrt{k}}{1-\gamma/\sqrt{k}}\right)|R| \\
&= (1+\gamma/\sqrt{k})(1+\gamma/\sqrt{k}+(\gamma/\sqrt{k})^2+(\gamma/\sqrt{k})^3+\cdots)|R| \\
&\qquad\qquad \left(\frac{1}{1-x} = 1 + x + x^2 + \cdots\right) \\
&\leq (1+\gamma/\sqrt{k})(1+2\gamma/\sqrt{k})|R| \qquad \text{(as } \gamma/\sqrt{k} \leq 1/2\text{)} \\
&= (1+3\gamma/\sqrt{k}+2(\gamma/\sqrt{k})^2)|R| \\
&= (1+4\gamma/\sqrt{k})|R| \qquad \text{(as } \gamma/\sqrt{k} \leq 1/2\text{)} \\
&= (1+c/\sqrt{k})|R| \qquad \square
\end{aligned}$$

Summary

- Design a local search algorithm with parameter k.
- Consider the solution B returned by the algorithm and an optimal solution R.
- Set up a bipartite planar graph G with bipartition B and R.
- Find a k-partitioning of G into $\Theta(n/k)$ regions, each region consisting of at most k vertices, and the boundary composed of $O(\sqrt{k})$ vertices.
- Bound the size of B in terms of the size of R using the neighborhood relationships of internal blue vertices in each region.

Extensions: Maximization problems (see Aschner et al.), Max Coverage Problems with Cardinality Constraints (see Chaplick et al.), ...

References

References

1. Arya et al., Local search heuristics for k-median and facility location problems, SIAM Jl. Computing 33(3): 544-562, 2004.
2. A. Gupta and K. Tangwongsam, Simpler Analyses of Local Search Algorithms for Facility Location, `http://arxiv.org/abs/0809.2554`, 2008.
3. N.H. Mustafa and S. Ray, Improved results on geometric hitting set problems, Discrete and Computational Geometry 44:883-895, 2010.
4. R. Ravi, Lectures at MSR India Winter School , 2012.
5. A. Gupta, Lectures on Approximation Algorithms, 2005.
6. R. Aschner, M.J. Katz, G. Morgenstern and Y. Yuditsky, Approximation schemes for covering and packing, WALCOM, Lecture Notes in Computer Science 7748: 89-100, Springer, 2013.
7. S. Chaplick, M. De, A. Ravsky, and J. Spoerhase, Approximation Schemes for Geometric Coverage Problems, ESA, LIPIcs: 112: 17:1–17:15, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.