Local Search Based Approximation Algorithms

The k-median problem

Vinayaka Pandit

IBM India Research Laboratory

joint work with Naveen Garg, Rohit Khandekar, and Vijay Arya

Outline

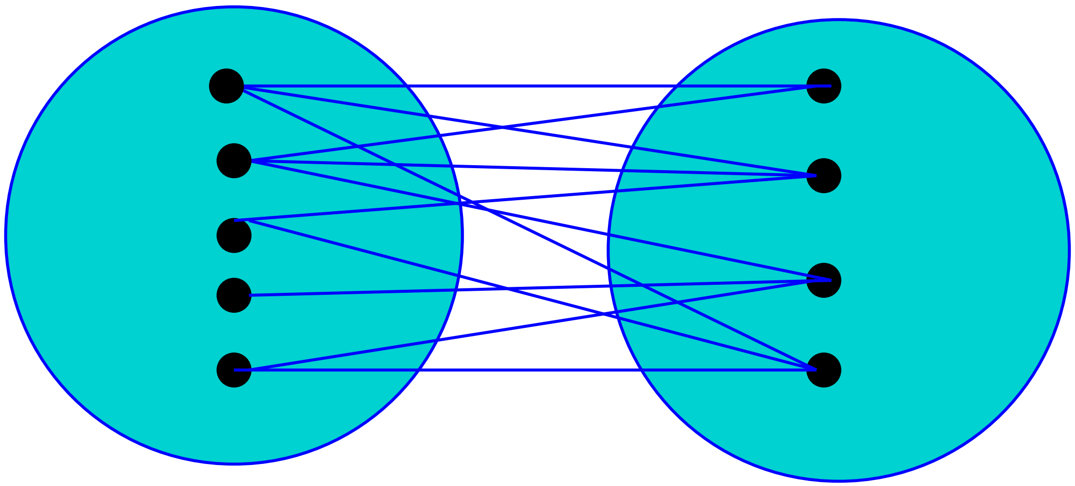
- Local Search Technique
- Simple Example Using Max-CUT
- Analysis of a popular heuristic for the k-median problem
- Some interesting questions

Local Search Technique

- Let $\mathcal{F}$ denote the set of all feasible solutions for a problem instance $\mathcal{I}$.
- Define a function $\mathcal{N} : \mathcal{F} \rightarrow 2^{\mathcal{F}}$ which associates for each solution, a set of neighboring solutions.
- Start with some feasible solution and iteratively perform "local operations". Suppose $S_C \in \mathcal{F}$ is the current solution. We move to any solution $S_N \in \mathcal{N}(S_C)$ which is strictly better than S_C.
- Output S_L, a locally optimal solution for which no solution in $\mathcal{N}(S_L)$ is strictly better than S_L itself.

Example: MAX-CUT

Given a graph $G = (V, E)$,



Partition V into A, B s.t. #edges between A and B is maximized.

Note that MAX-CUT is $\leq |E|$.

Local Search for MAX-CUT

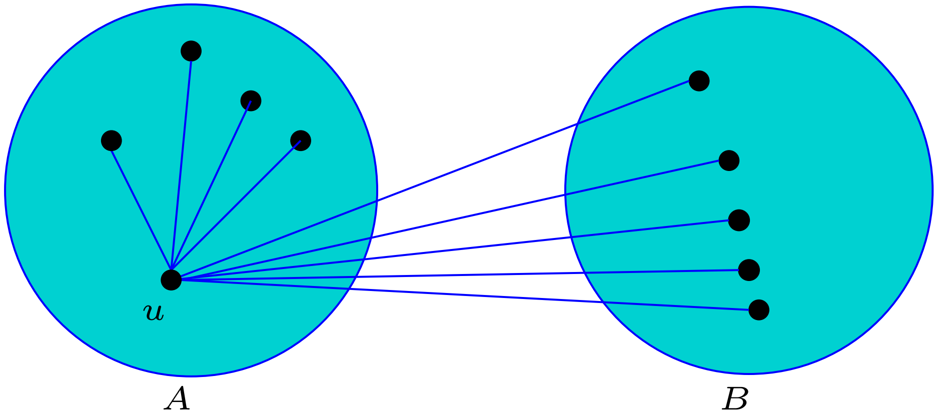
Algorithm `Local Search for MAX-CUT.`

1. $A, B \leftarrow$ any partition of V;
2. While $\exists \quad u \in V$ such that in-degree(u) > out-degree(u),
   do
      if($u \in A$), Move u to B
      else, Move u to A
   done
3. return A, B

Neighborhood Function

- Solution Space: the set of all partitions.
- Neighborhood Function: Neighbors of a partition (A, B) are all the partitions (A', B') obtained by interchanging the side of a single vertex.

Analysis for MAX-CUT



1. in-d(u) $\leq$ out-d(u) **(Apply Conditions for Local Optimality)**

2. $\sum_{u \in V}$in-d(u) $\leq \sum_{u \in V}$out-d(u) **(Consider suitable set of local operations)**

3. #Internal Edges $\leq$ #Cut Edges

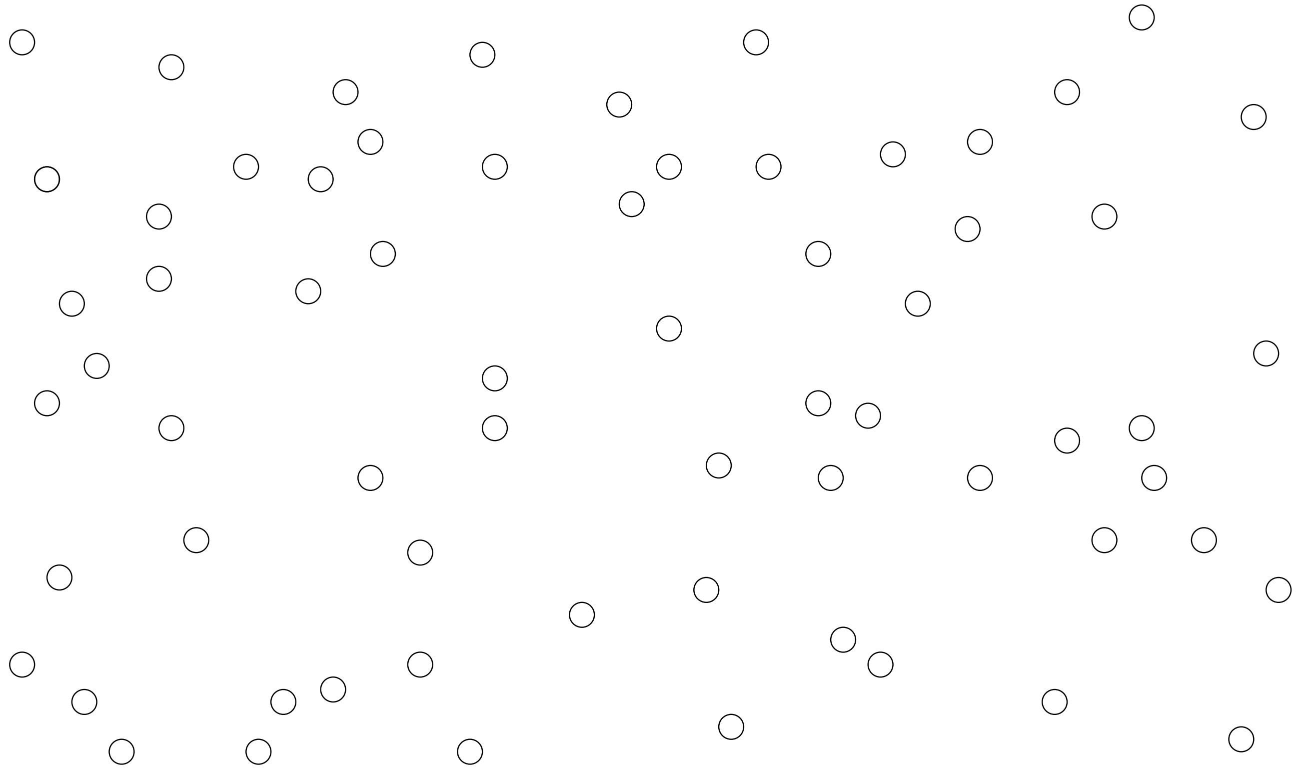
#Cut-edges $\geq \frac{|E|}{2} \Rightarrow 2$-approximation **(Infer)**

Local Search for Approximation

- Folklore: the 2-approx for MAX-CUT
- Fürer and Raghavachari: Additive Approx for Min. Degree Spanning Tree.
- Lu and Ravi: Constant Factor approx for Spanning Trees with maximum leaves.
- Könemann and Ravi: Bi-criteria approximation for Bounded Degree MST.
- Quite successful as a technique for Facility Location and Clustering problems. Started with Korupolu et. al.

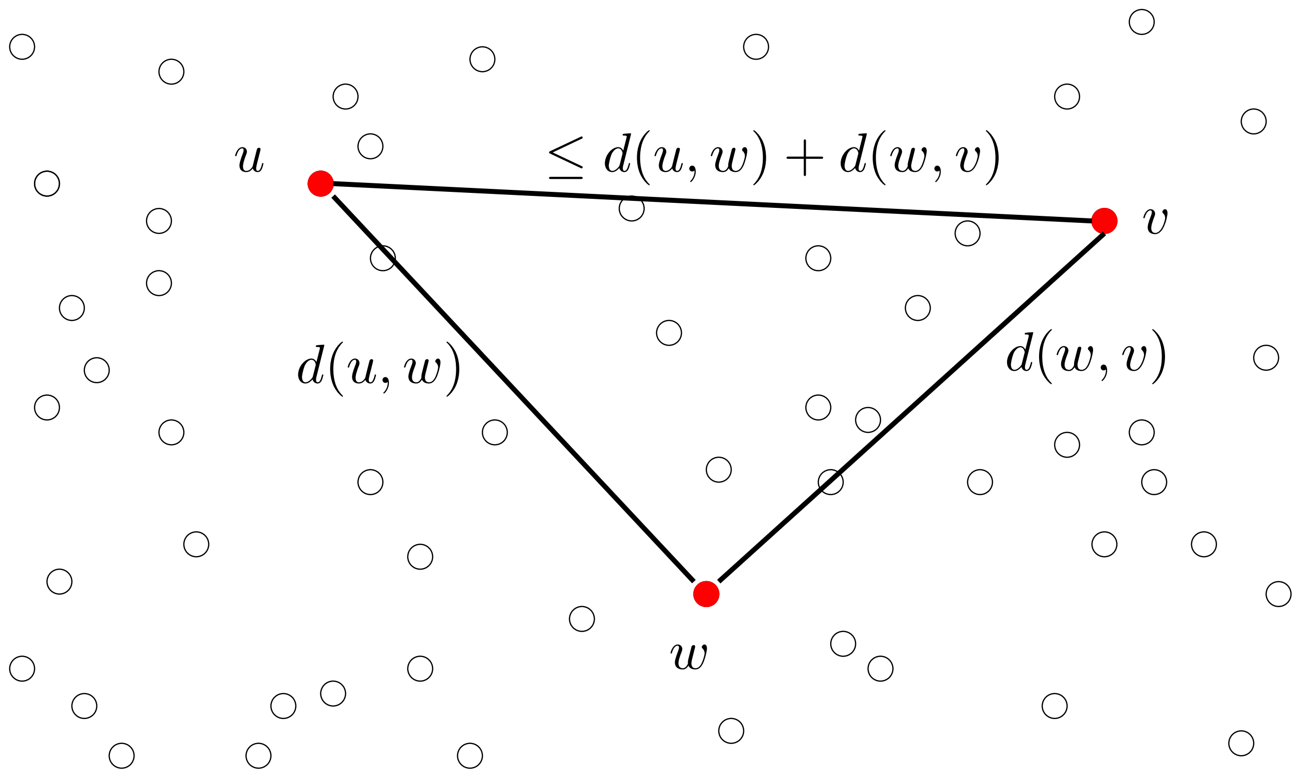
The k-median problem

We are given n points in a *metric space*.



The k-median problem

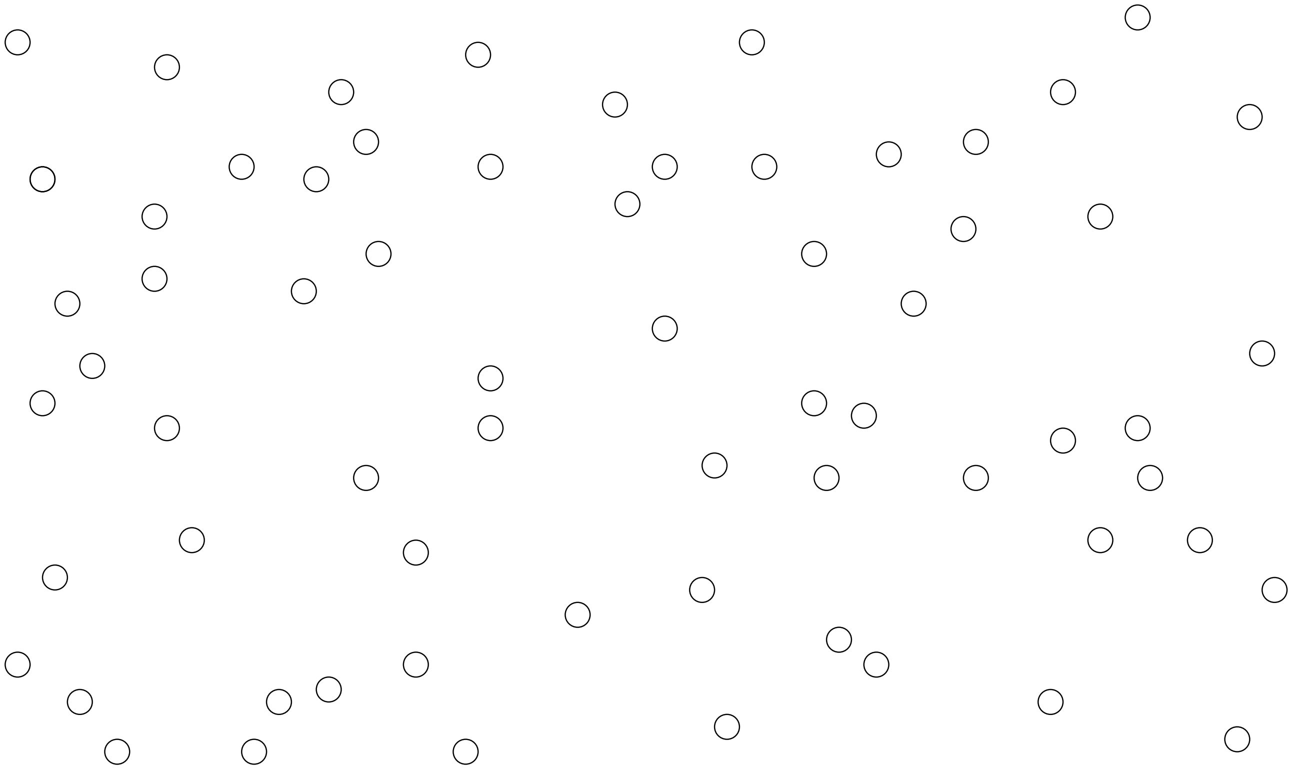
We are given n points in a *metric space*.



$$d(u, v) \geq 0, \qquad d(u, u) = 0, \qquad d(u, v) = d(v, u)$$

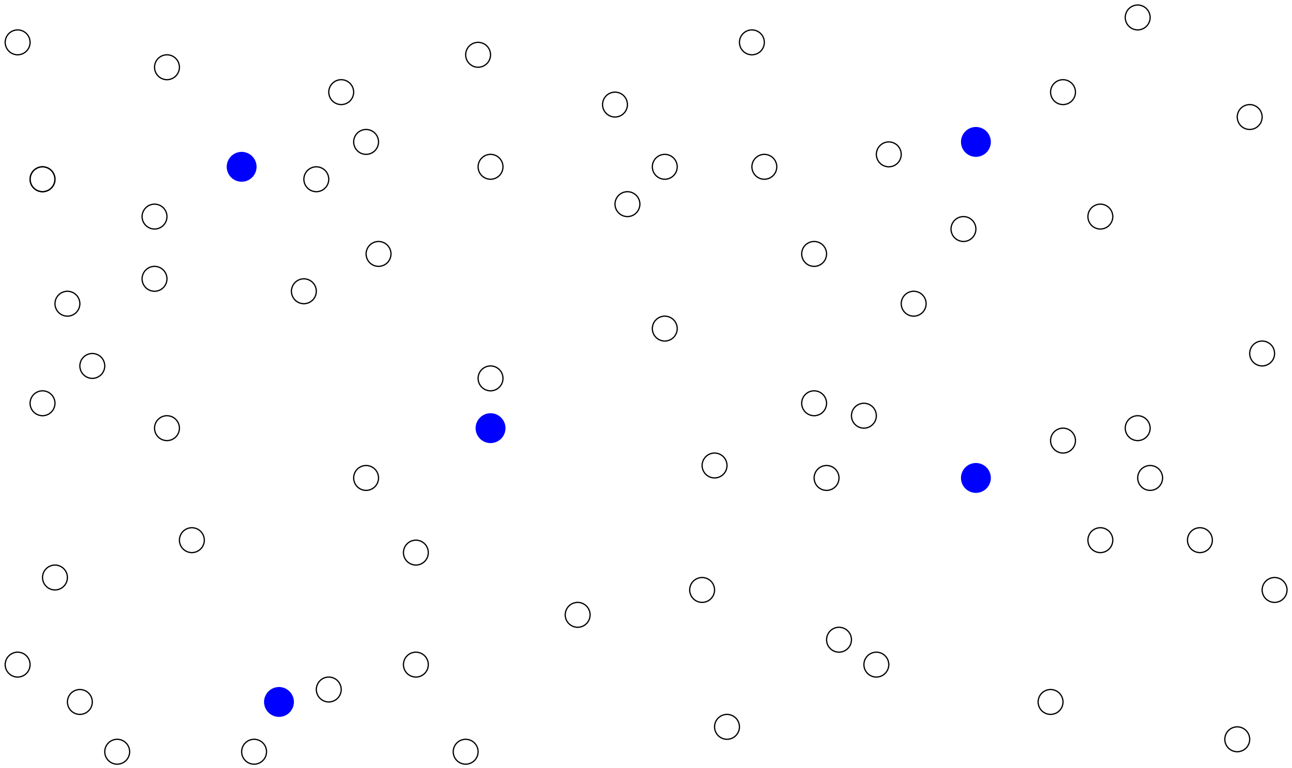
The k-median problem

We are given n points in a *metric space*.



The k-median problem

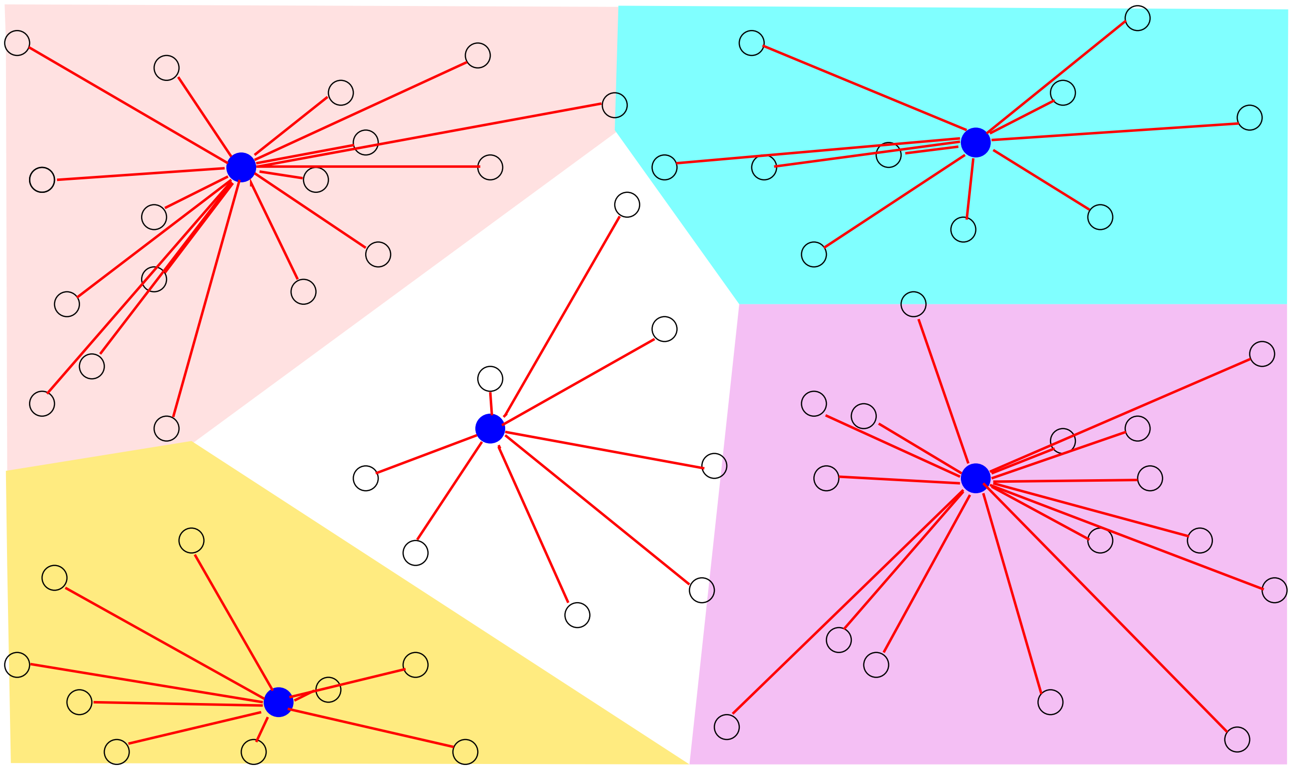
We are given n points in a *metric space*.



We want to identify k “medians” such that the sum of distances of all the points to their nearest medians is minimized.

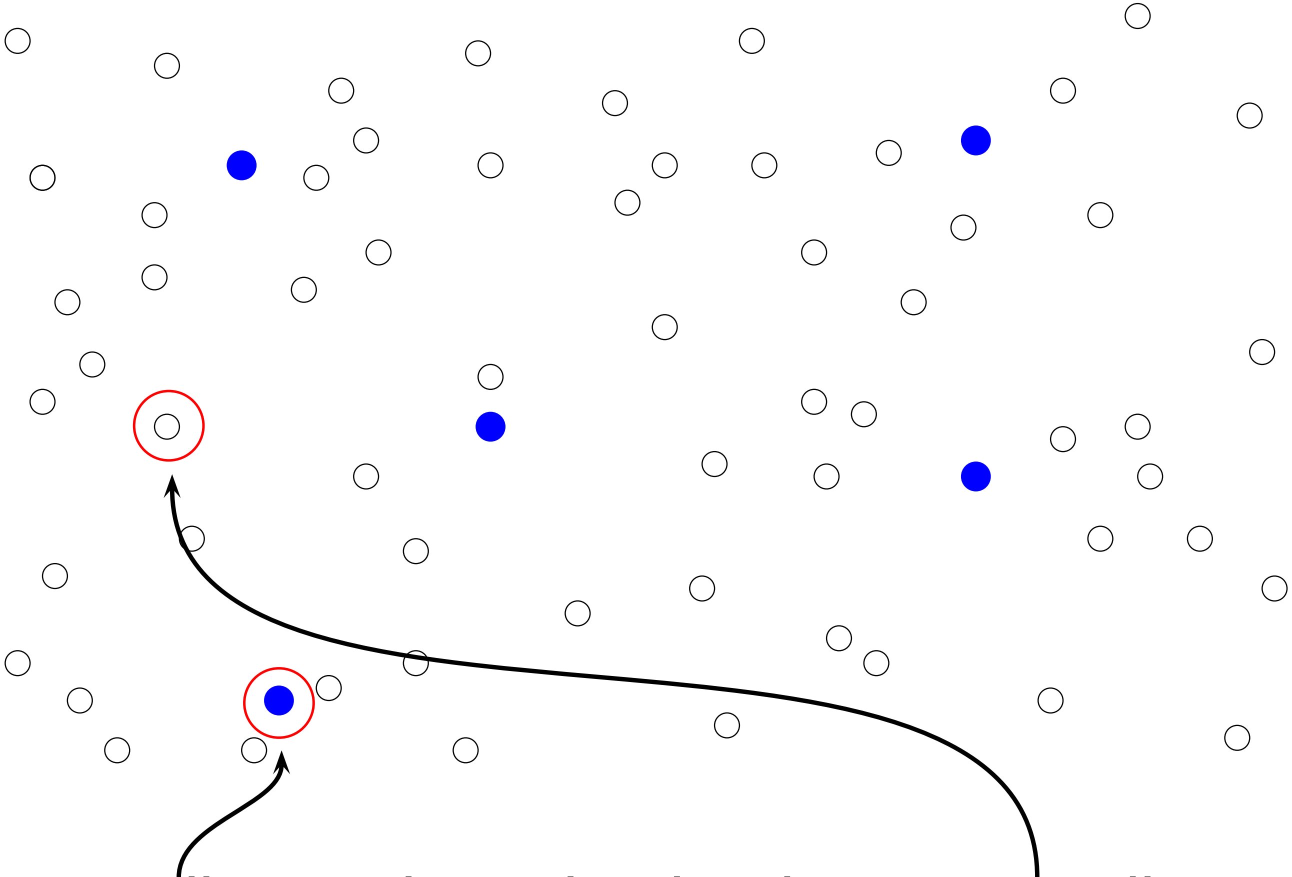
The k-median problem

We are given n points in a *metric space*.



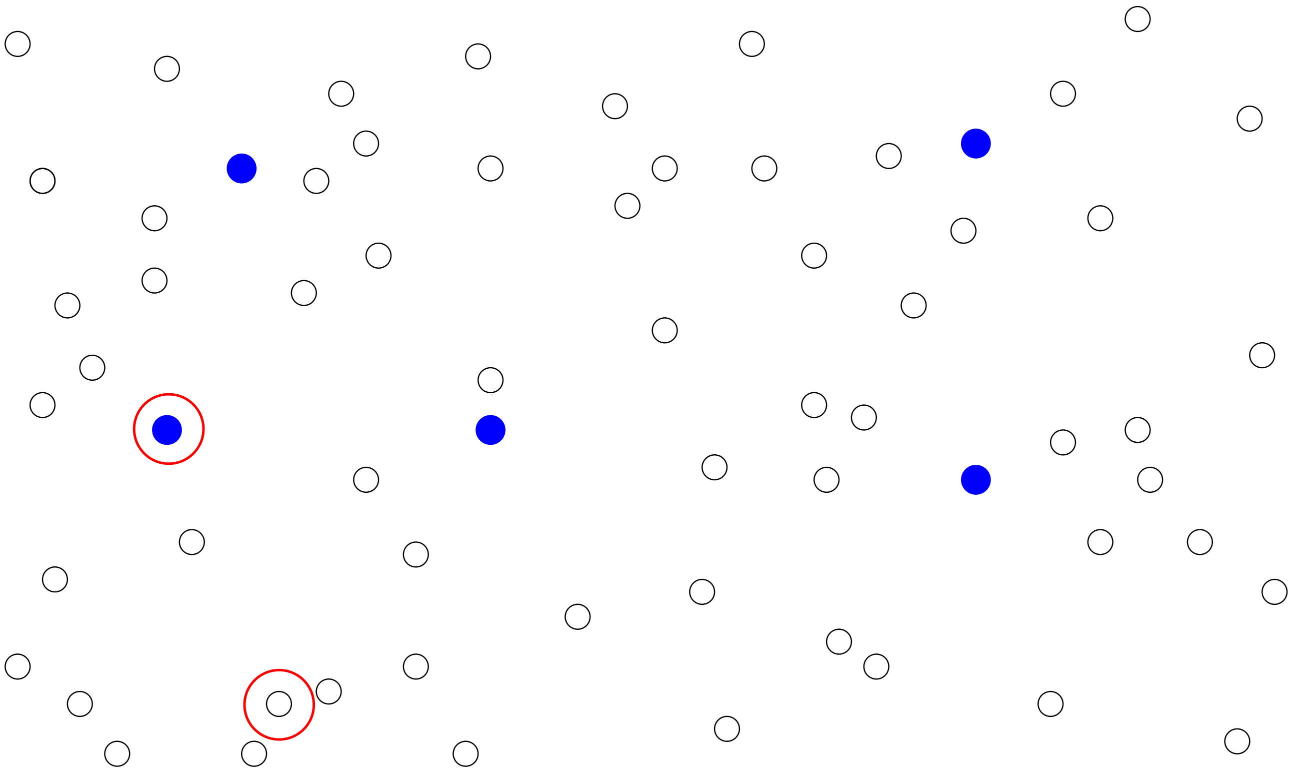
We want to identify k “medians” such that the sum of lengths of all the red segments is minimized.

A local search algorithm



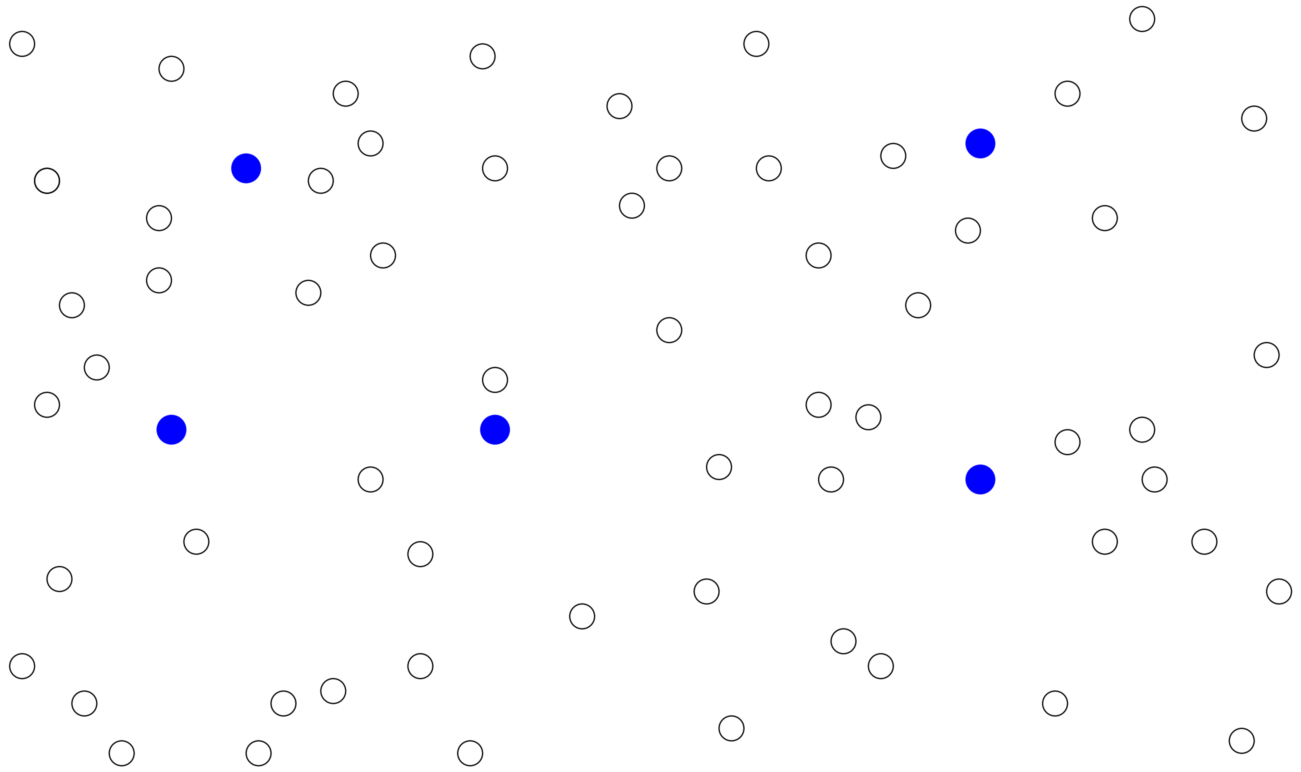
Identify a median and a point that is not a median.

A local search algorithm



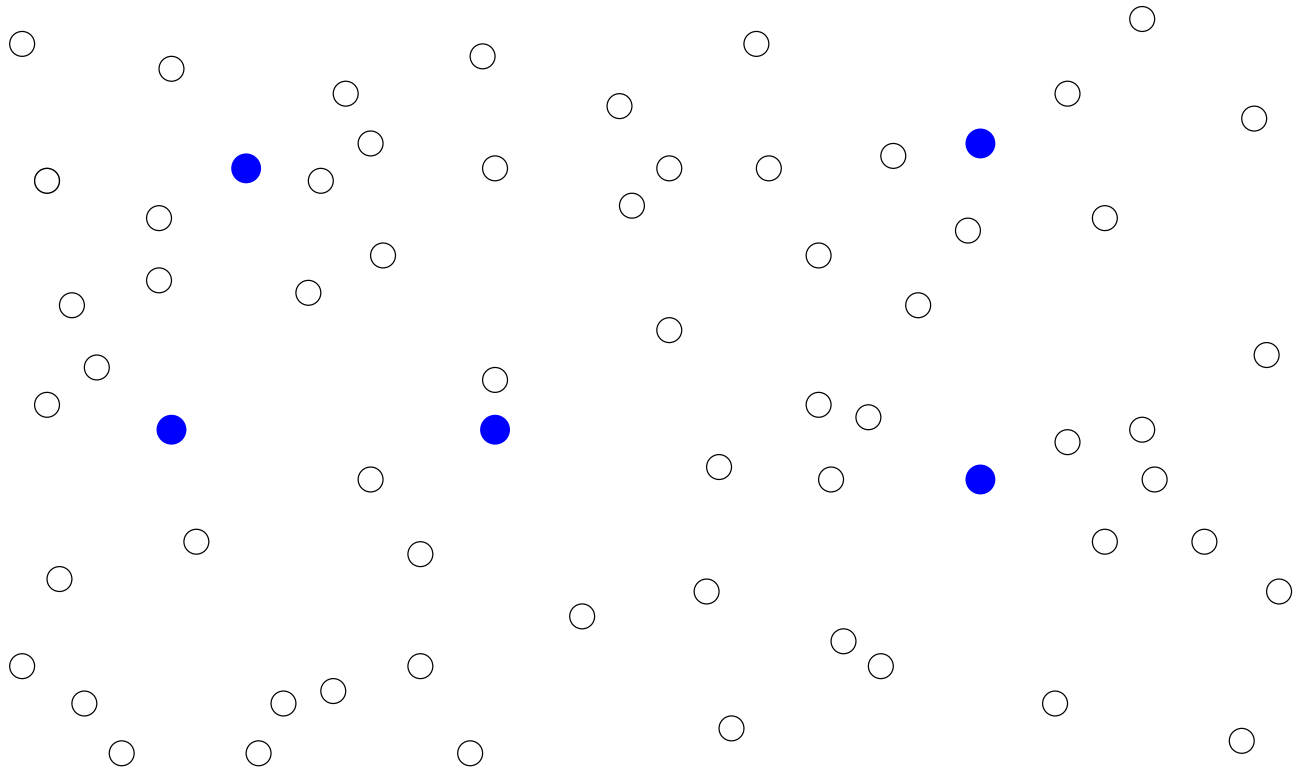
And SWAP tentatively!

A local search algorithm



Perform the swap, only if the new solution is “better” (has less cost) than the previous solution.

A local search algorithm



Perform the swap, only if the new solution is “better” (has less cost) than the previous solution.

Stop, if there is no swap that improves the solution.

The algorithm

Algorithm `Local Search.`

1. $S \leftarrow$ any k medians
2. While $\exists \;\; s \in S$ and $s' \notin S$ such that,
   $$cost(S - s + s') < cost(S),$$
   do $S \leftarrow S - s + s'$
3. return S

The algorithm

Algorithm `Local Search.`

1. $S \leftarrow$ any k medians
2. While $\exists \;\; s \in S$ and $s' \notin S$ such that,

   $cost(S - s + s') < (1 - \epsilon)cost(S)$,

   do $S \leftarrow S - s + s'$
3. return S

Main theorem

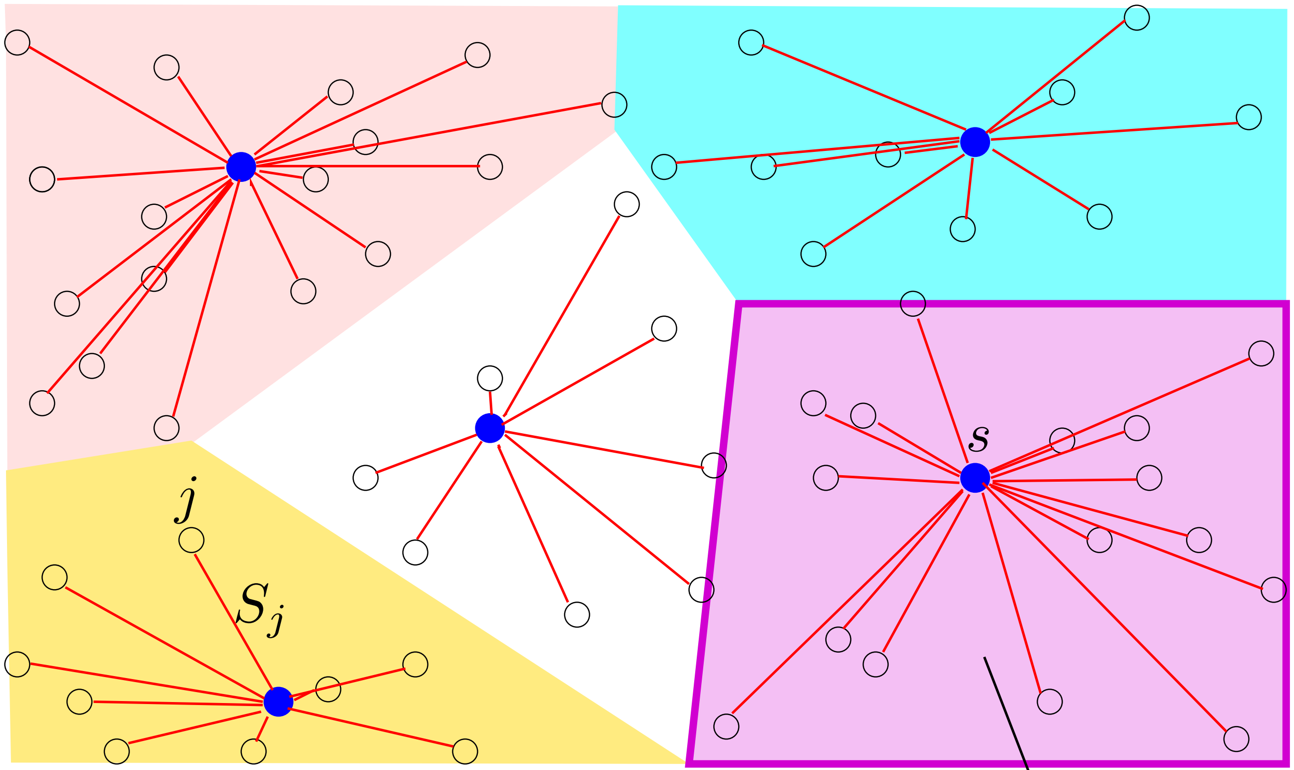
The local search algorithm described above computes a solution with cost (the sum of distances) at most 5 times the minimum cost.

Main theorem

The local search algorithm described above computes a solution with cost (the sum of distances) at most 5 times the minimum cost.

Korupolu, Plaxton, and Rajaraman (1998) analyzed a variant in which they permitted adding, deleting, and swapping medians and got $(3 + 5/\epsilon)$ approximation by taking $k(1+\epsilon)$ medians.

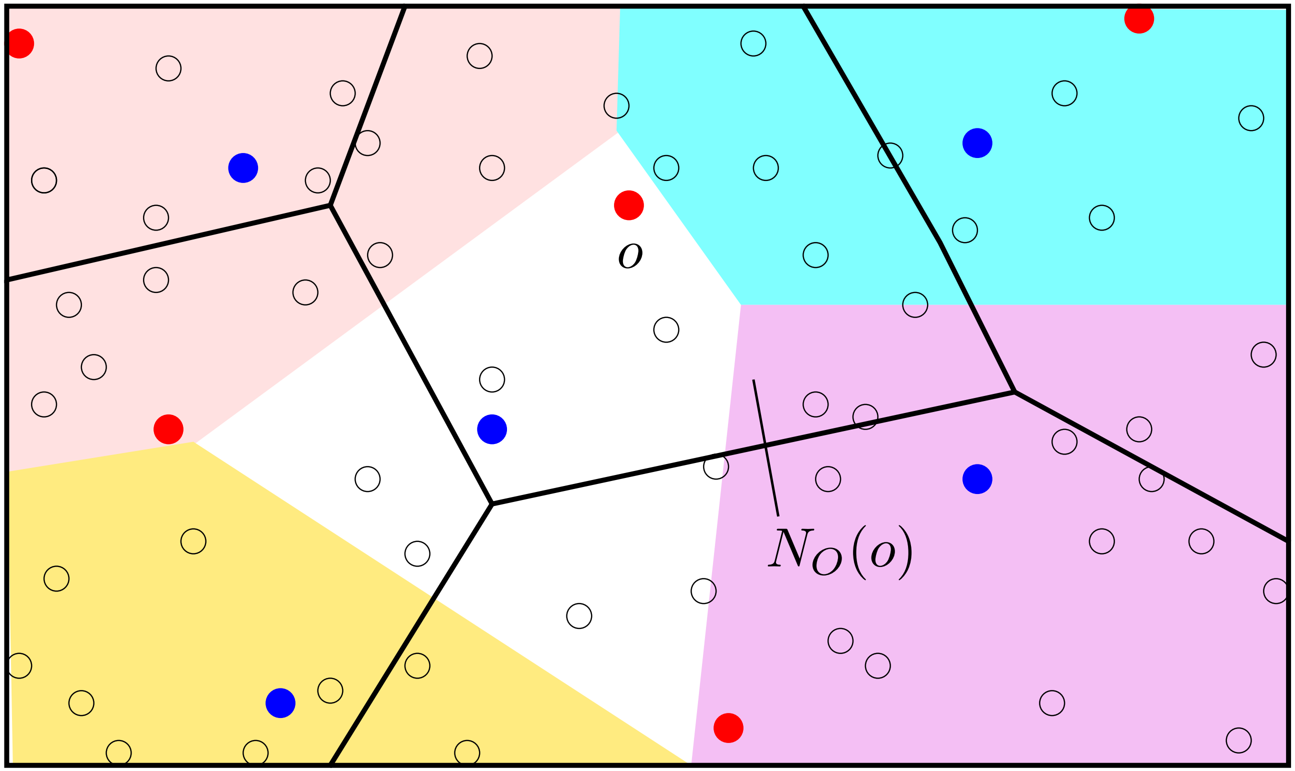
Some notation



$S = \{\bullet\ \bullet\ \bullet\ \bullet\ \bullet\}$ $\qquad |S| = k$ $\qquad N_S(s)$

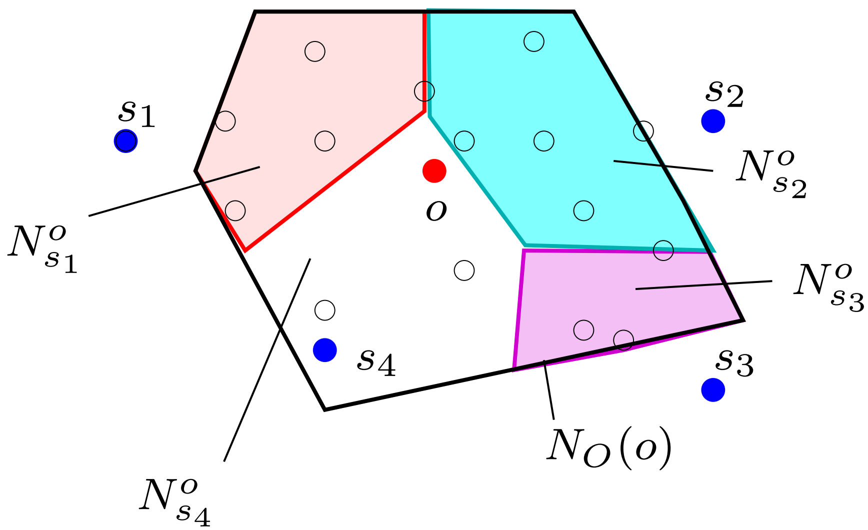
$cost(S)$ = the sum of lengths of all the red segments

Some more notation



$O = \{$ ● ● ● ● ● $\}$ $\qquad |O| = k$

Some more notation



$$N^o_s = N_O(o) \cap N_S(s)$$

Local optimality of S

- Since S is a local optimum solution,

Local optimality of S

- Since S is a local optimum solution,

  We have,

$$cost(S - s + o) \ \geq \ cost(S) \qquad \text{for all } s \in S, o \in O.$$

Local optimality of S

- Since S is a local optimum solution,

  We have,

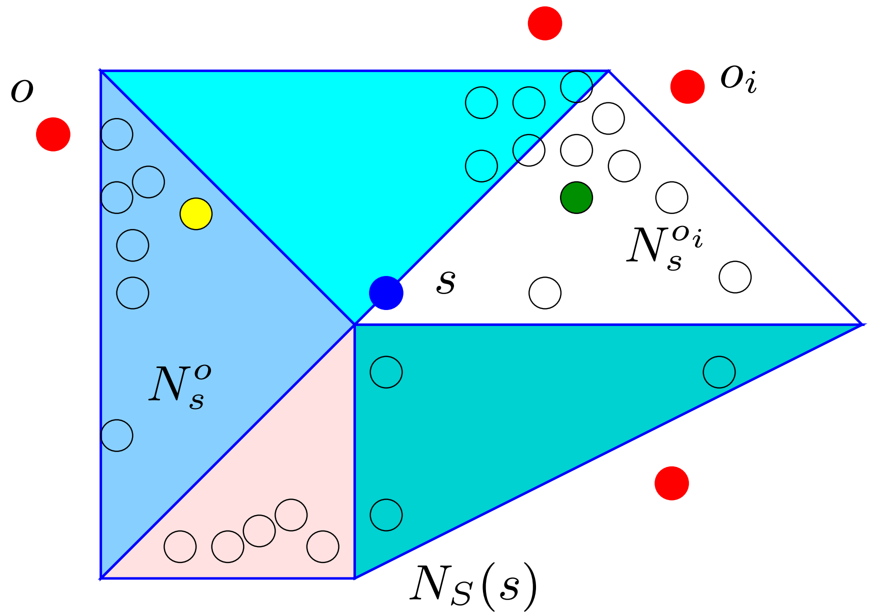
$$cost(S - s + o) \;\geq\; cost(S) \qquad \text{for all } s \in S, o \in O.$$

- We shall add k of these inequalities (chosen carefully) to show that,

$$cost(S) \;\leq\; 5 \cdot cost(O)$$

> >

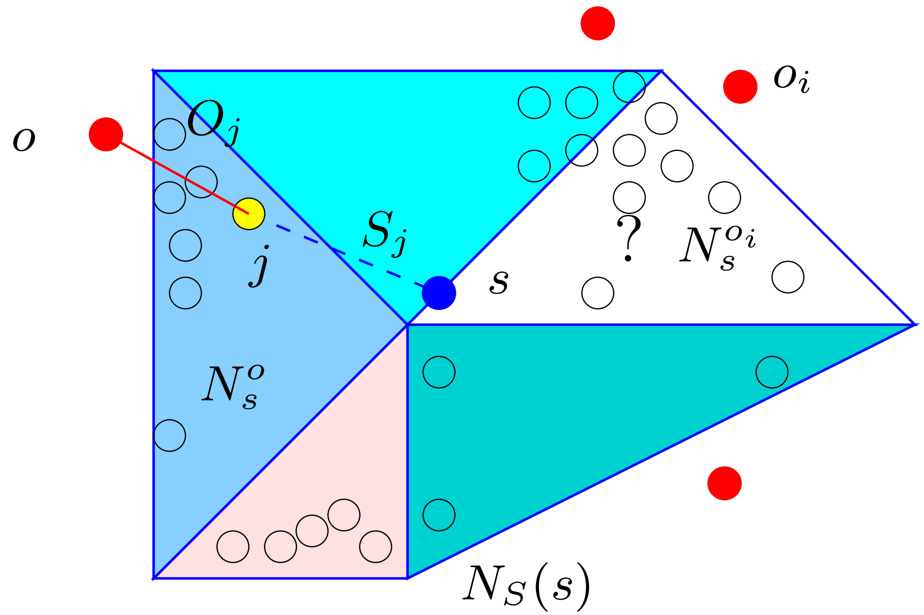
What happens when we swap <s, o>?



All the points in $N_S(s)$ have to be rerouted to one of the facilities in $S - \{s\} + \{o\}$.

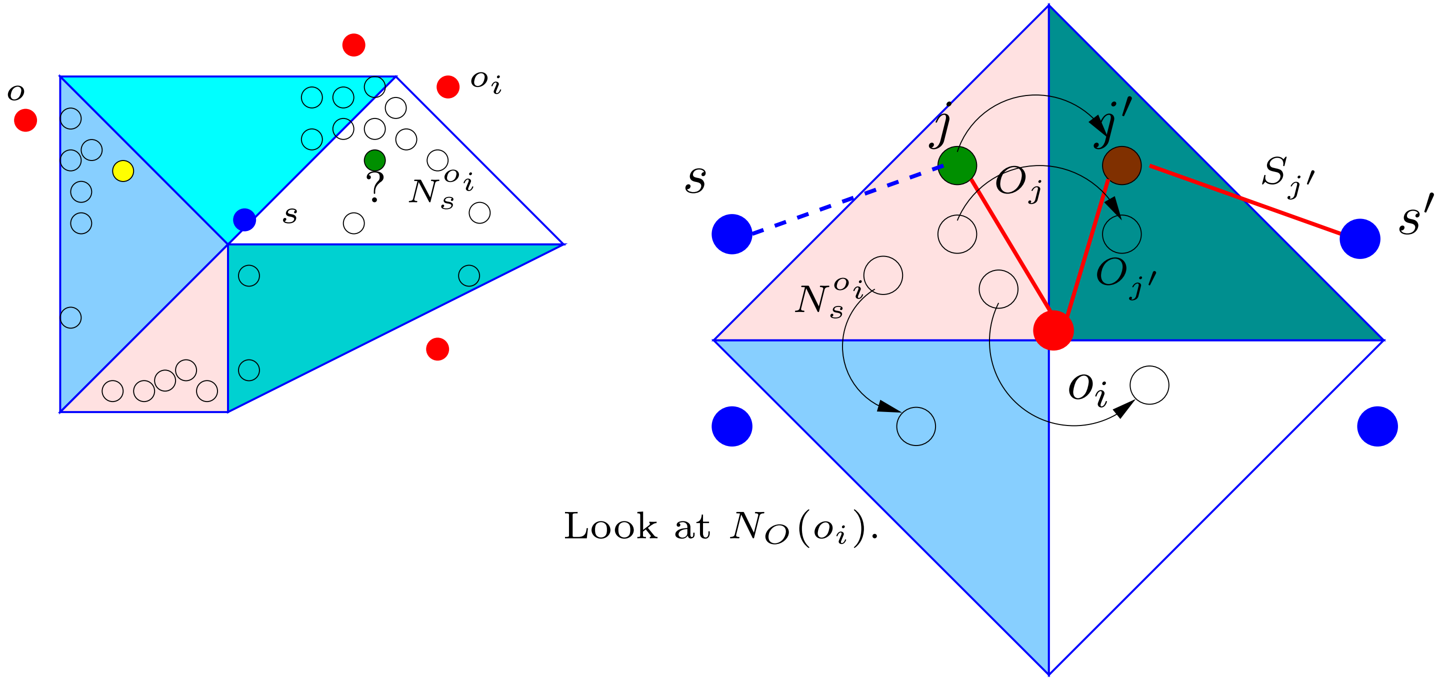
We are interested two types of clients: those belonging to N_s^o and those not belonging to N_s^o.

Rerouting $j \in N_s^o$



Rerouting is easy. Send it to o. Change in cost = $O_j - S_j$.

Rerouting $j \notin N_s^o$

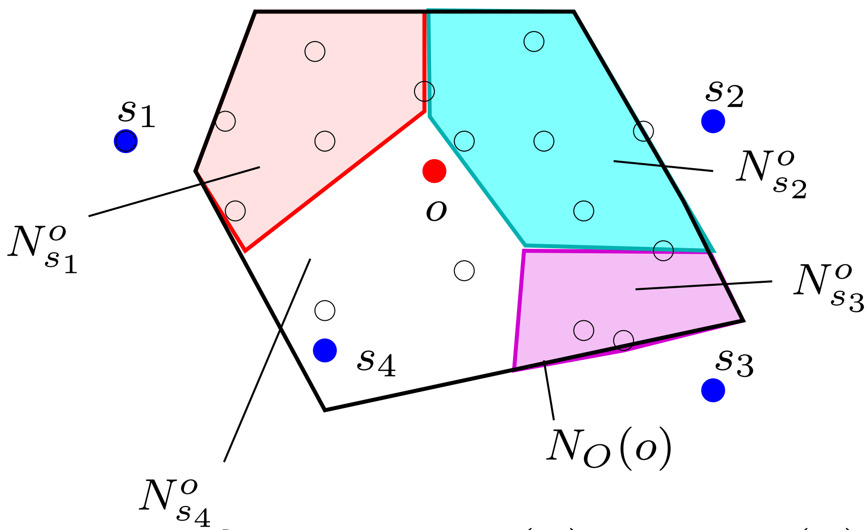


Map j to a unique $j' \in N_O(o_i)$ outside $N_s^{o_i}$ and route via j'.
Change in cost = $O_j + O_{j'} + S_{j'} - S_j$.

Ensure that every client is involved in exactly one reroute.

Therefore, the mapping need to be one-to-one and onto.

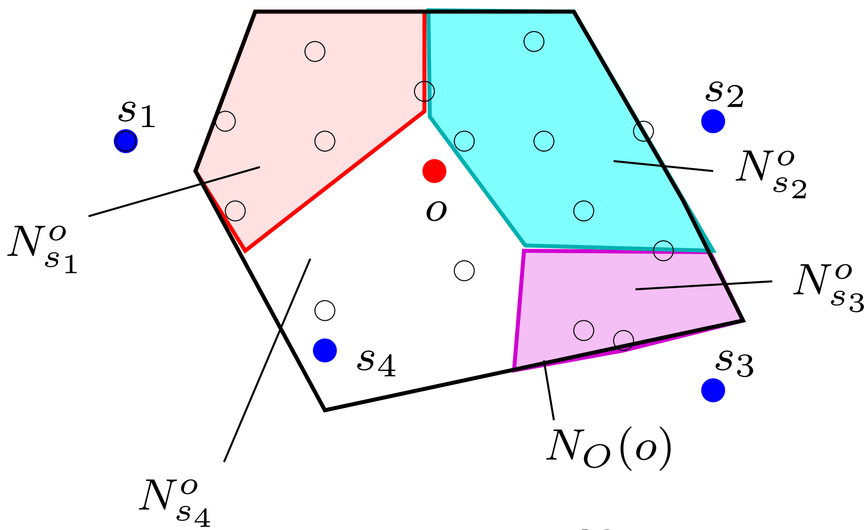
Desired mapping of clients inside $N_O(o)$



We desire a permutation $\pi : N_O(o) \rightarrow N_O(o)$ that satisfies the following property:

Client $j \in N^o_s$ should get mapped to $j' \in N_O(o)$, but outside N^o_s.

Notion of Capture



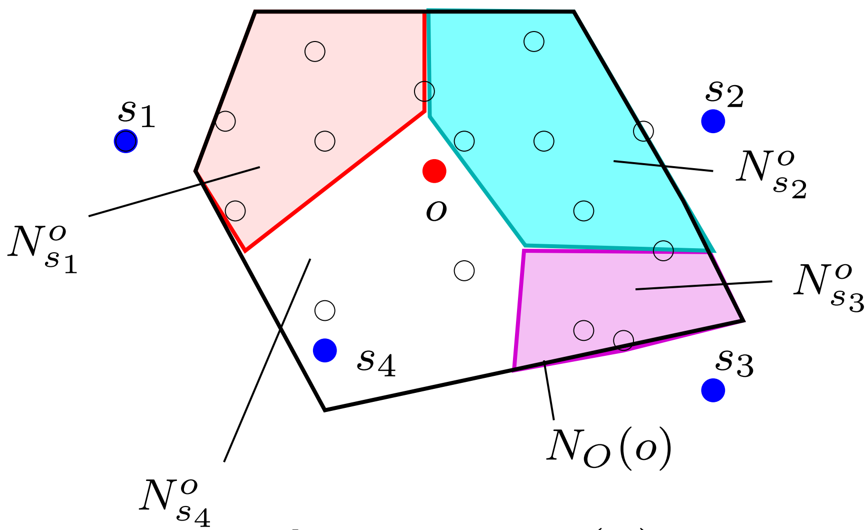
We say that $s \in S$ captures $o \in O$ if

$$|N^o_s| > \frac{|N_O(o)|}{2}.$$

Note: A facility $o \in O$ is captured precisely when a mapping as we described is not feasible.

Capture graph

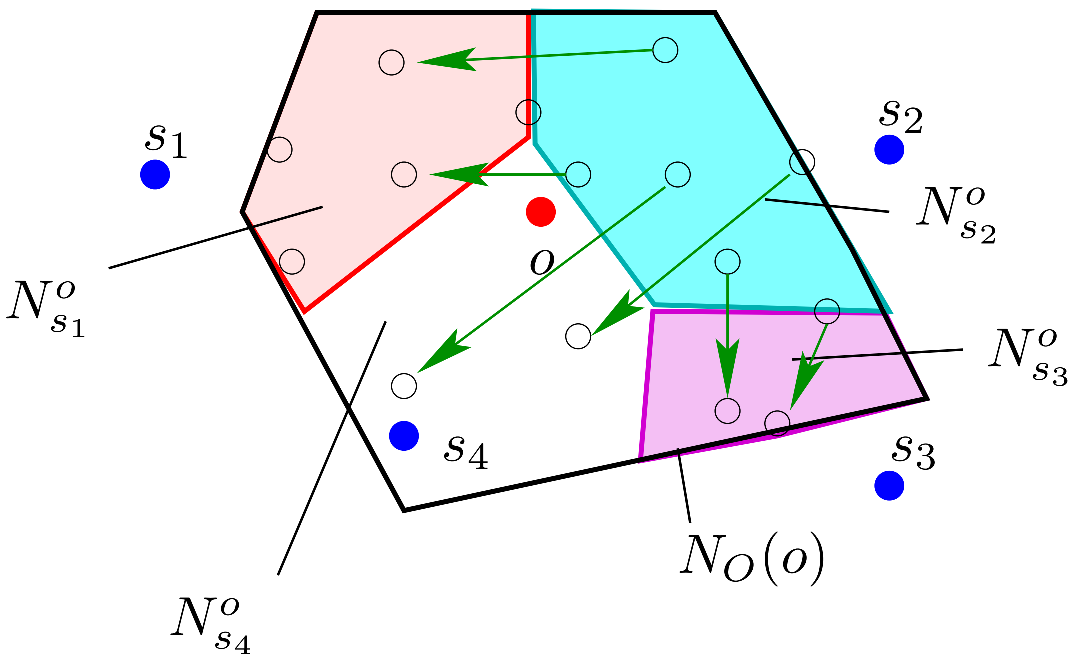
A mapping π



We consider a permutation $\pi : N_O(o) \rightarrow N_O(o)$ that satisfies the following property:

if s does not capture o then a point $j \in N^o_s$ should get mapped outside N^o_s.

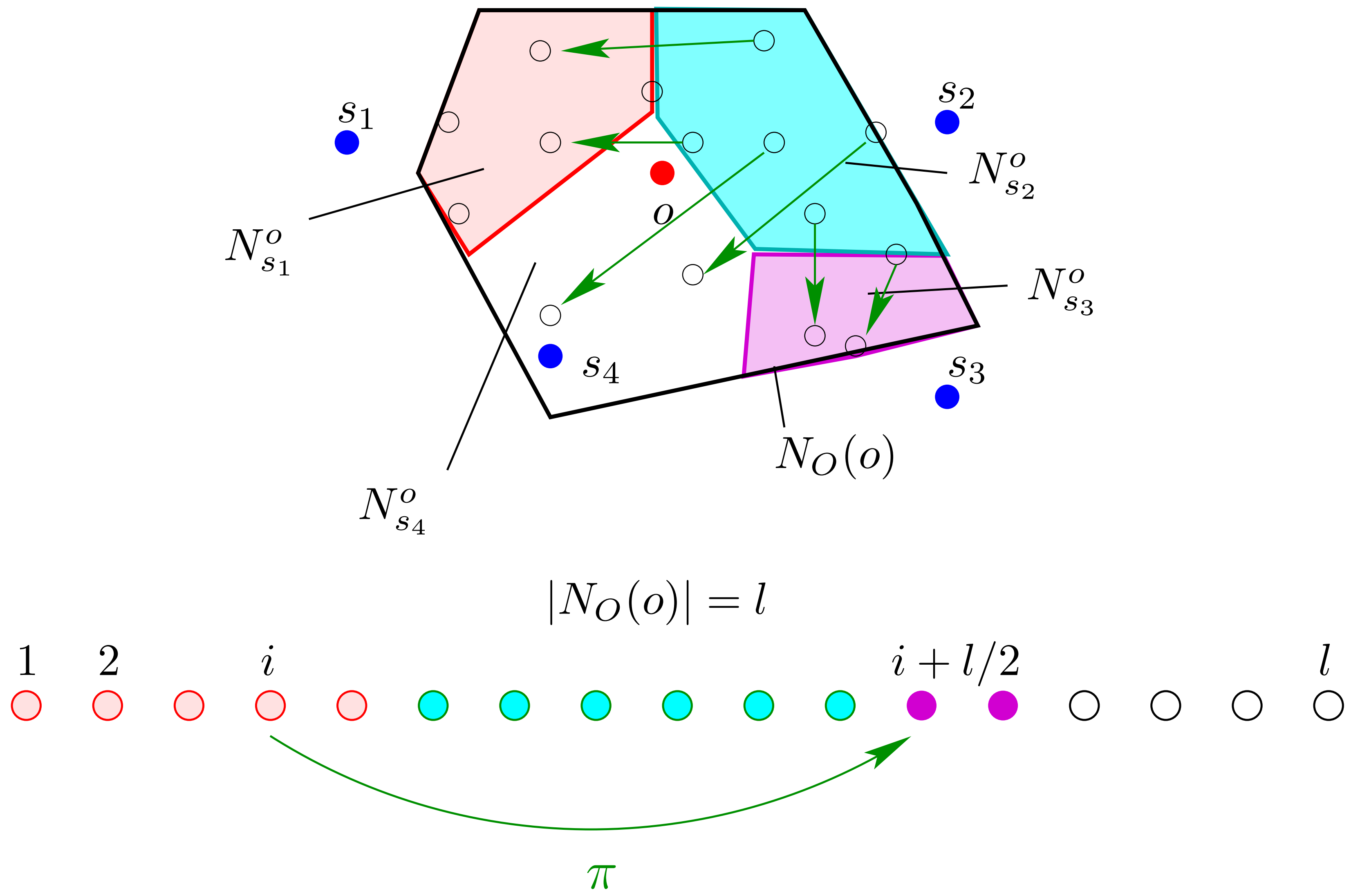
A mapping π



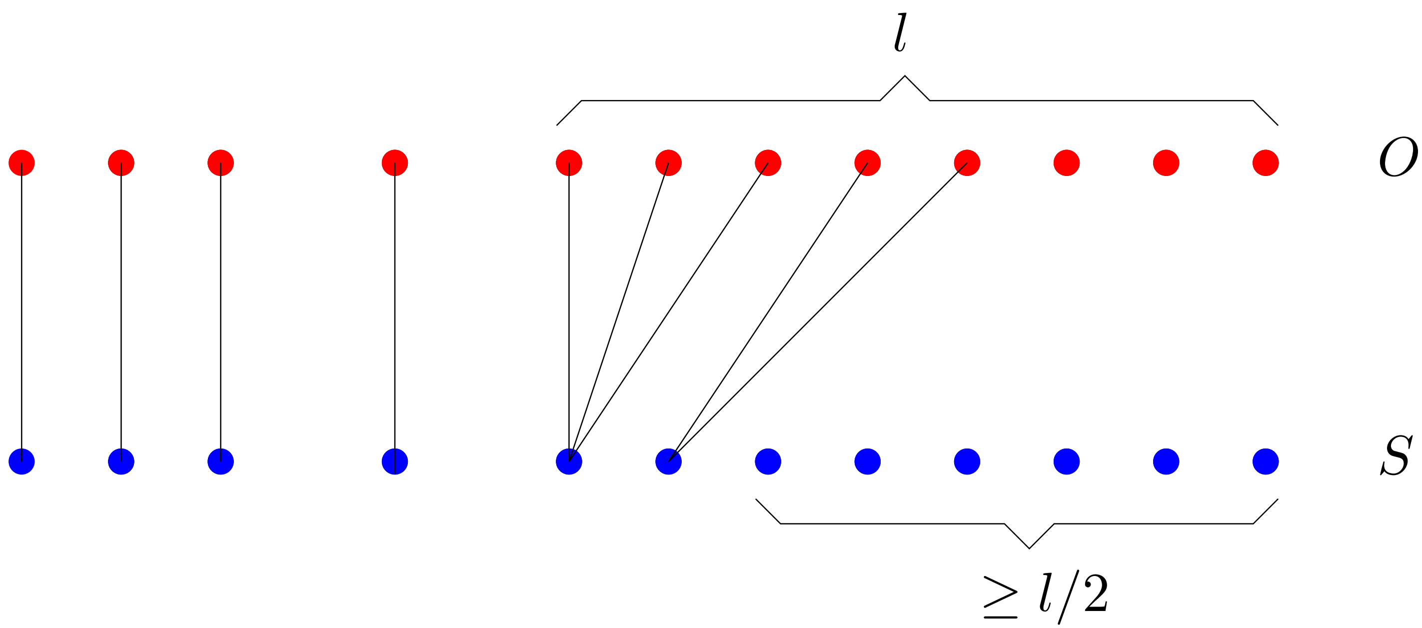
We consider a permutation $\pi : N_O(o) \rightarrow N_O(o)$ that satisfies the following property:

if s does not capture o then a point $j \in N^o_s$ should get mapped outside N^o_s.

A mapping π



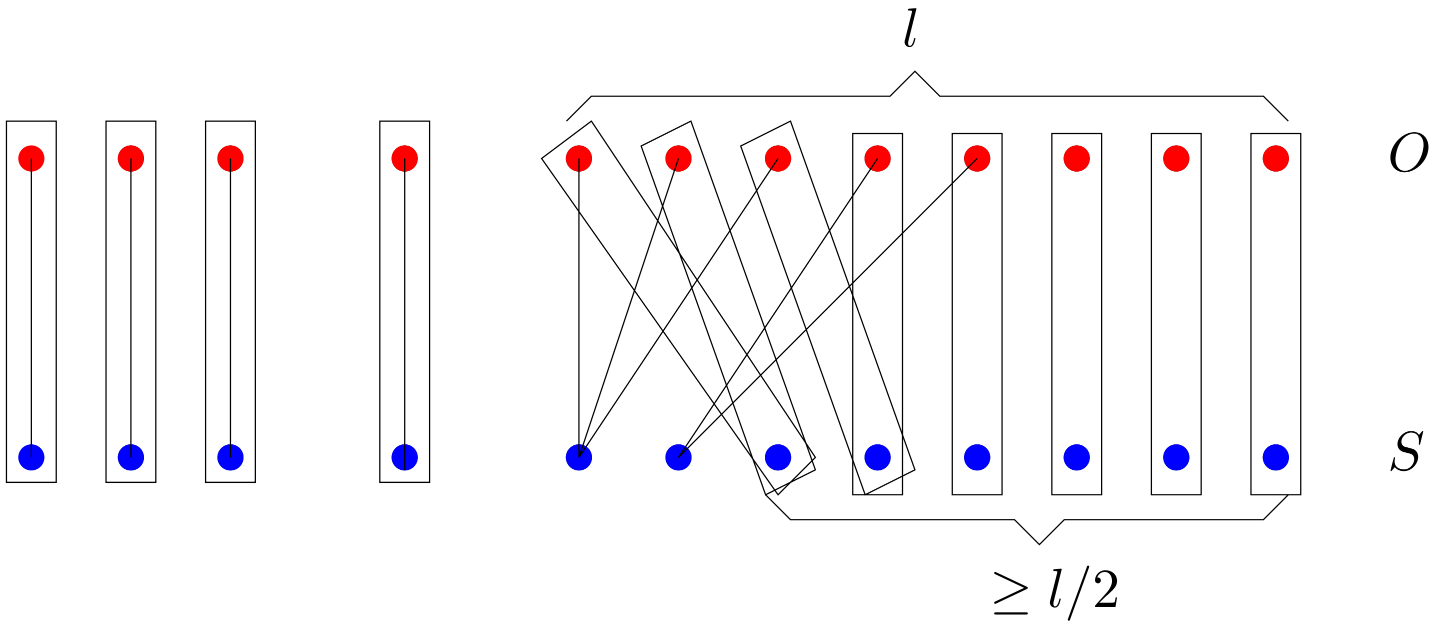
Capture graph



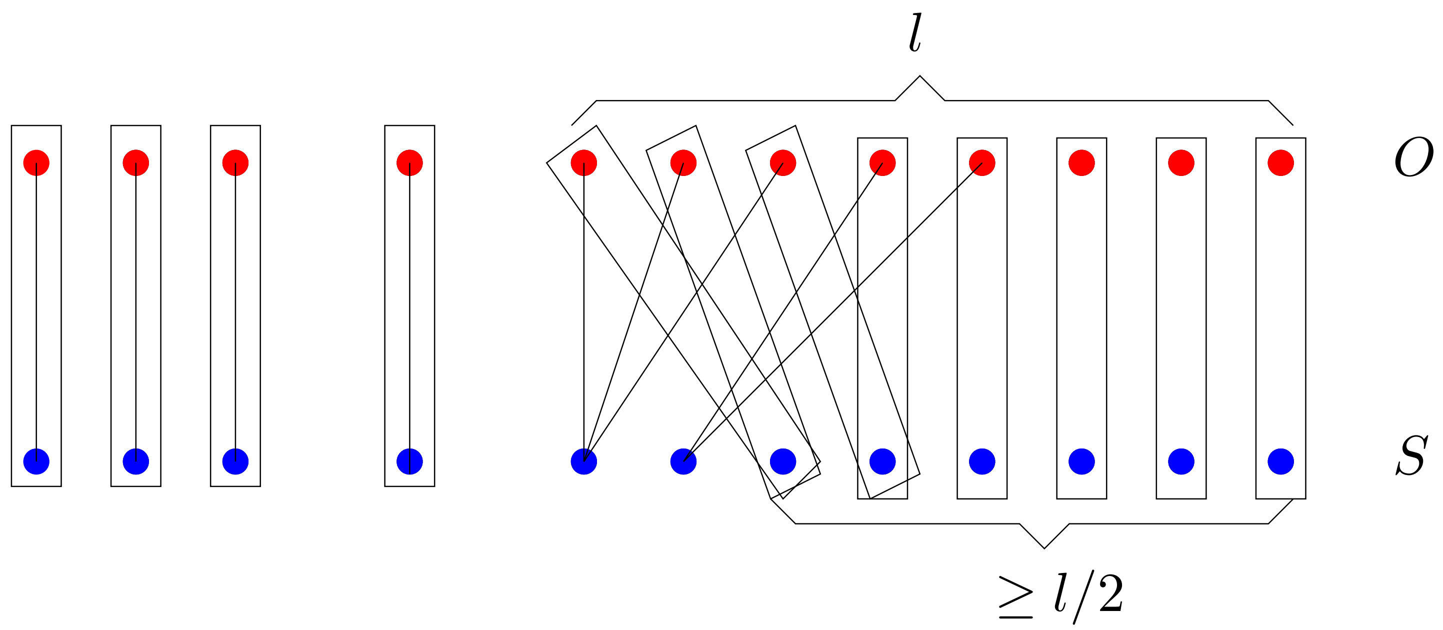
Construct a bipartite graph $G = (O, S, E)$ where there is an edge (o, s) if and only if $s \in S$ captures $o \in O$.

Capture

Swaps considered

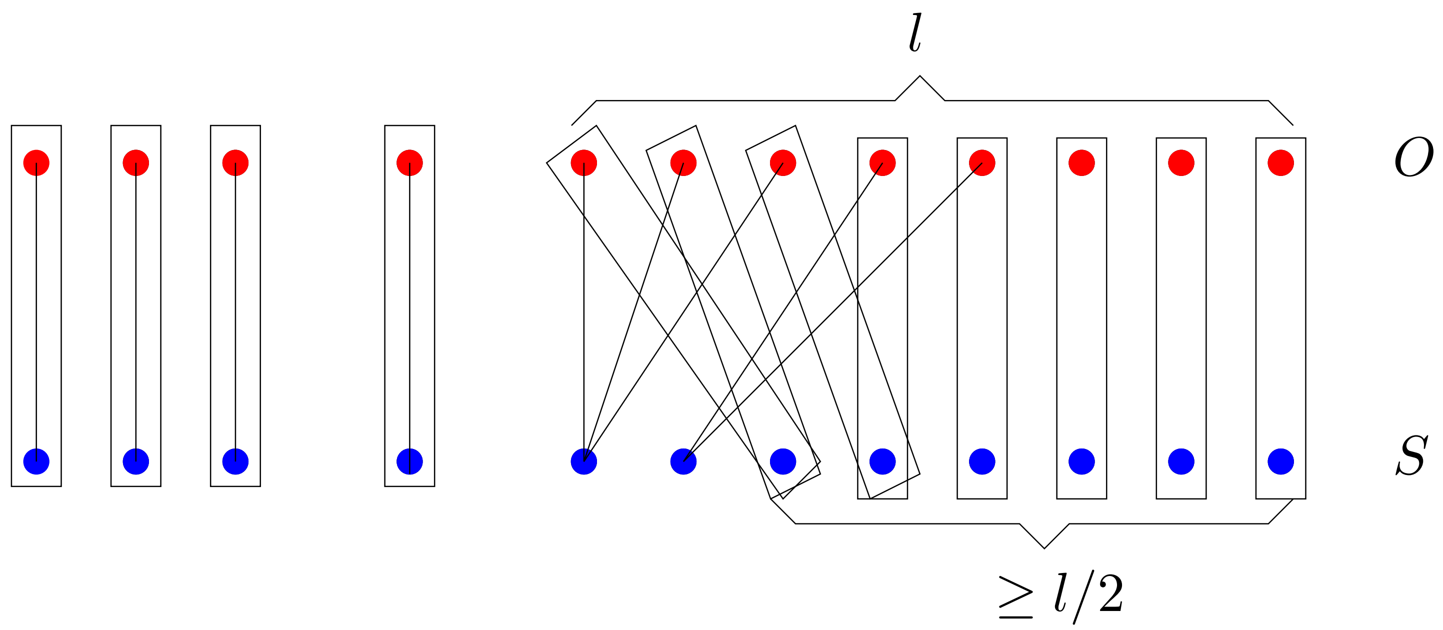


Swaps considered



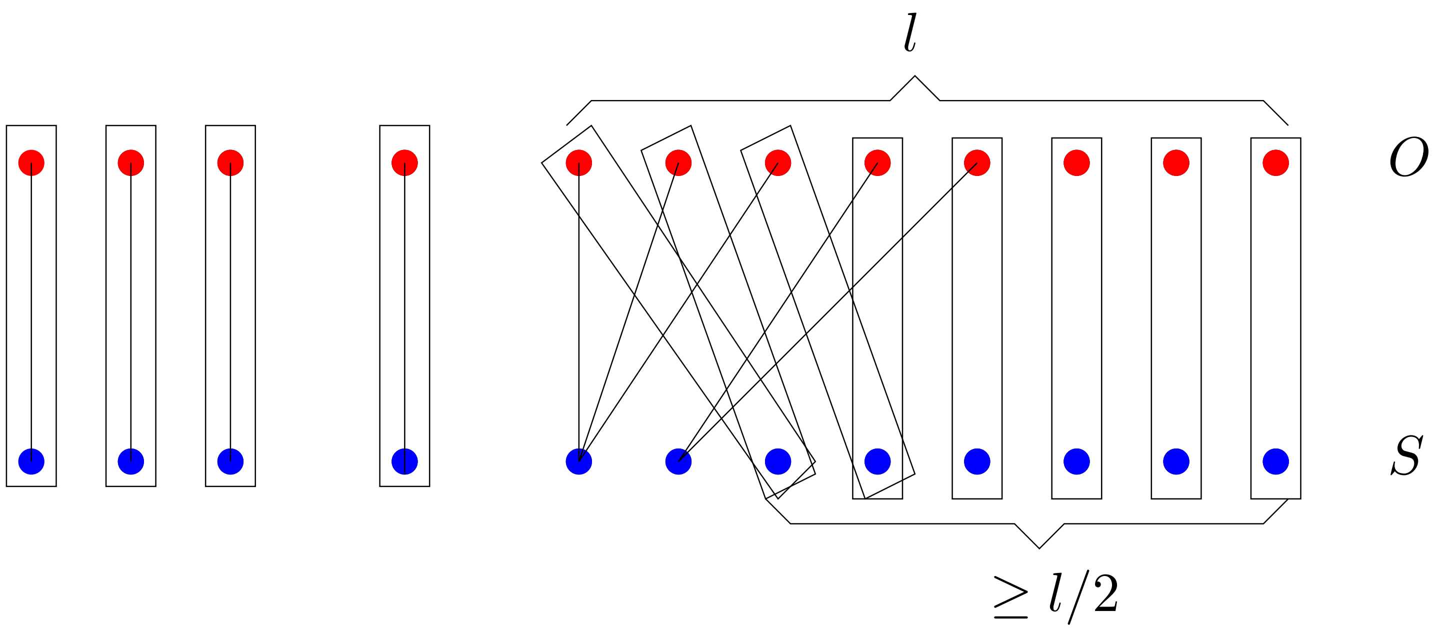
"Why consider the swaps?"

Properties of the swaps considered



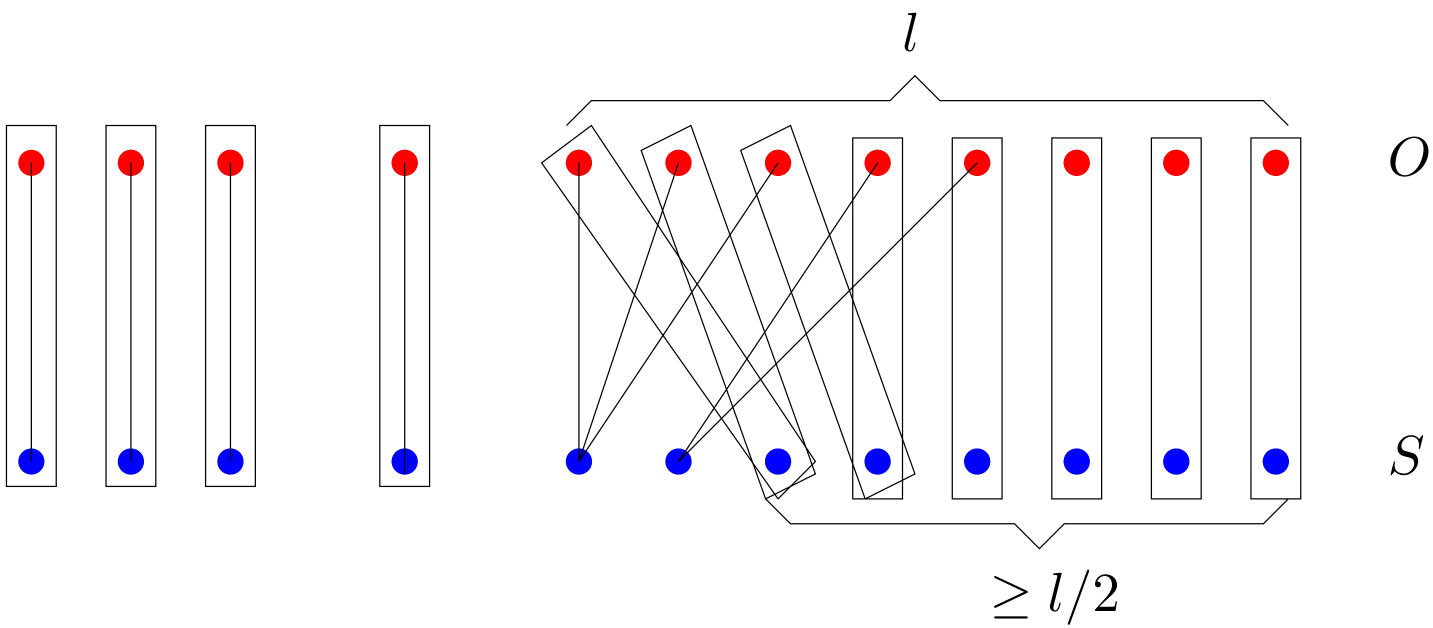
- If $\langle s, o \rangle$ is considered, then s does not capture any $o' \neq o$.

Properties of the swaps considered



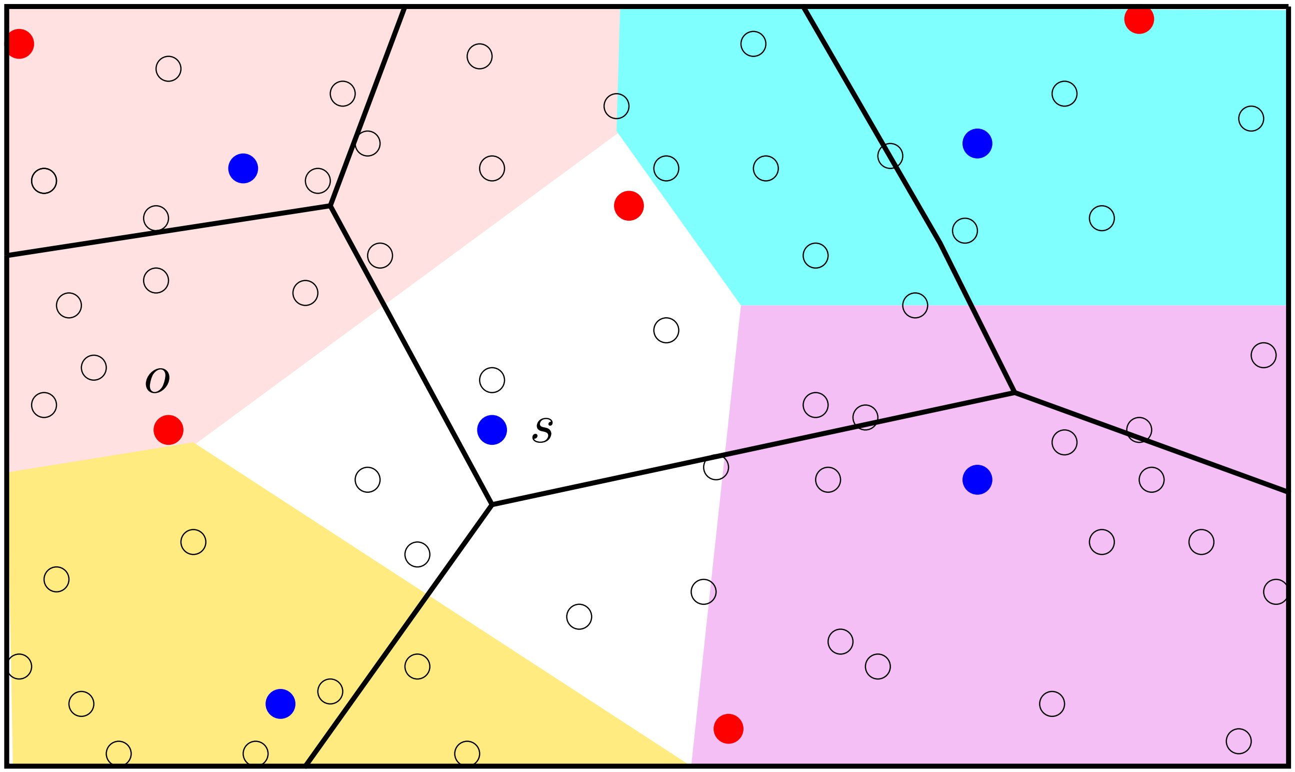
- If $\langle s, o \rangle$ is considered, then s does not capture any $o' \neq o$.
- Any $o \in O$ is considered in exactly one swap.

Properties of the swaps considered



- If $\langle s, o \rangle$ is considered, then s does not capture any $o' \neq o$.
- Any $o \in O$ is considered in exactly one swap.
- Any $s \in S$ is considered in at most 2 swaps.

Focus on a swap $\langle s, o \rangle$



Consider a swap $\langle s, o \rangle$ that is one of the k swaps defined above. We know $cost(S - s + o) \geq cost(S)$.

Upper bound on $cost(S - s + o)$

- In the solution $S - s + o$, each point is connected to the closest median in $S - s + o$.

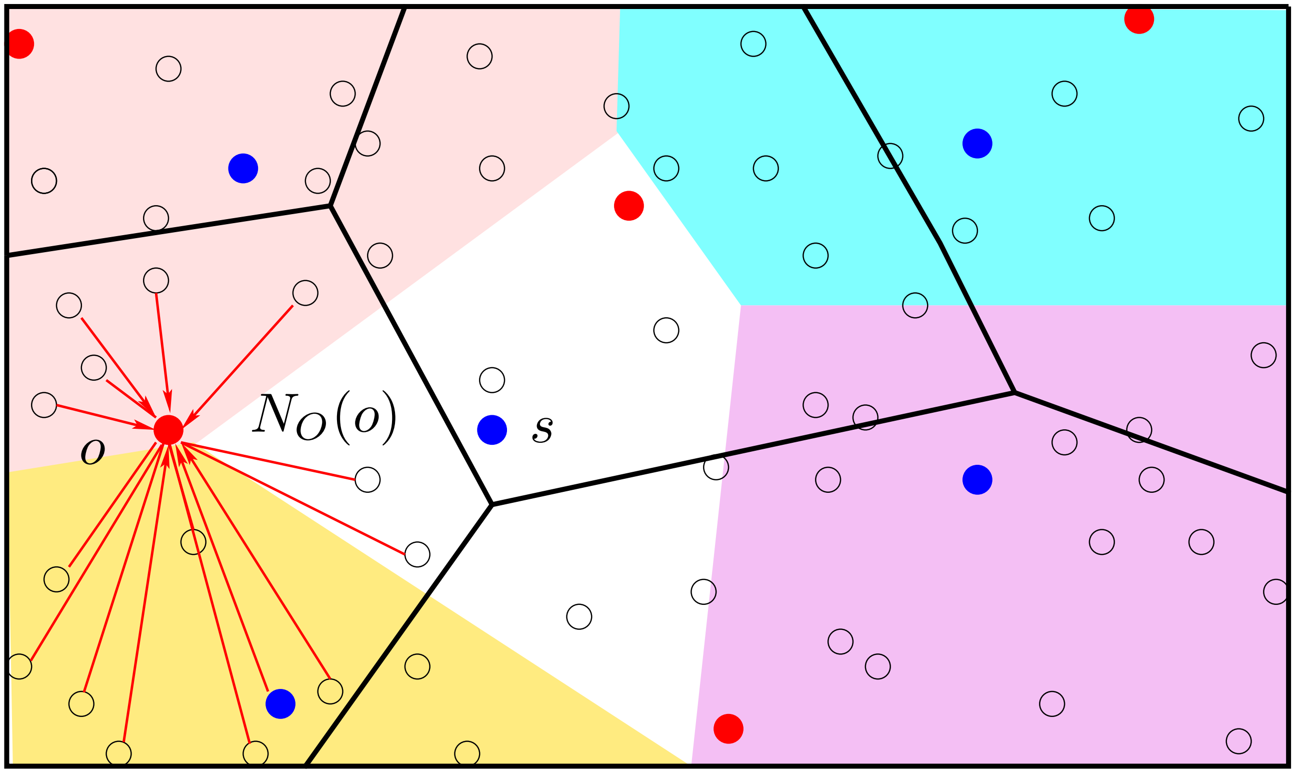
Upper bound on $cost(S - s + o)$

- In the solution $S - s + o$, each point is connected to the closest median in $S - s + o$.
- $cost(S - s + o)$ is the sum of distances of all the points to their nearest medians.

Upper bound on $cost(S - s + o)$

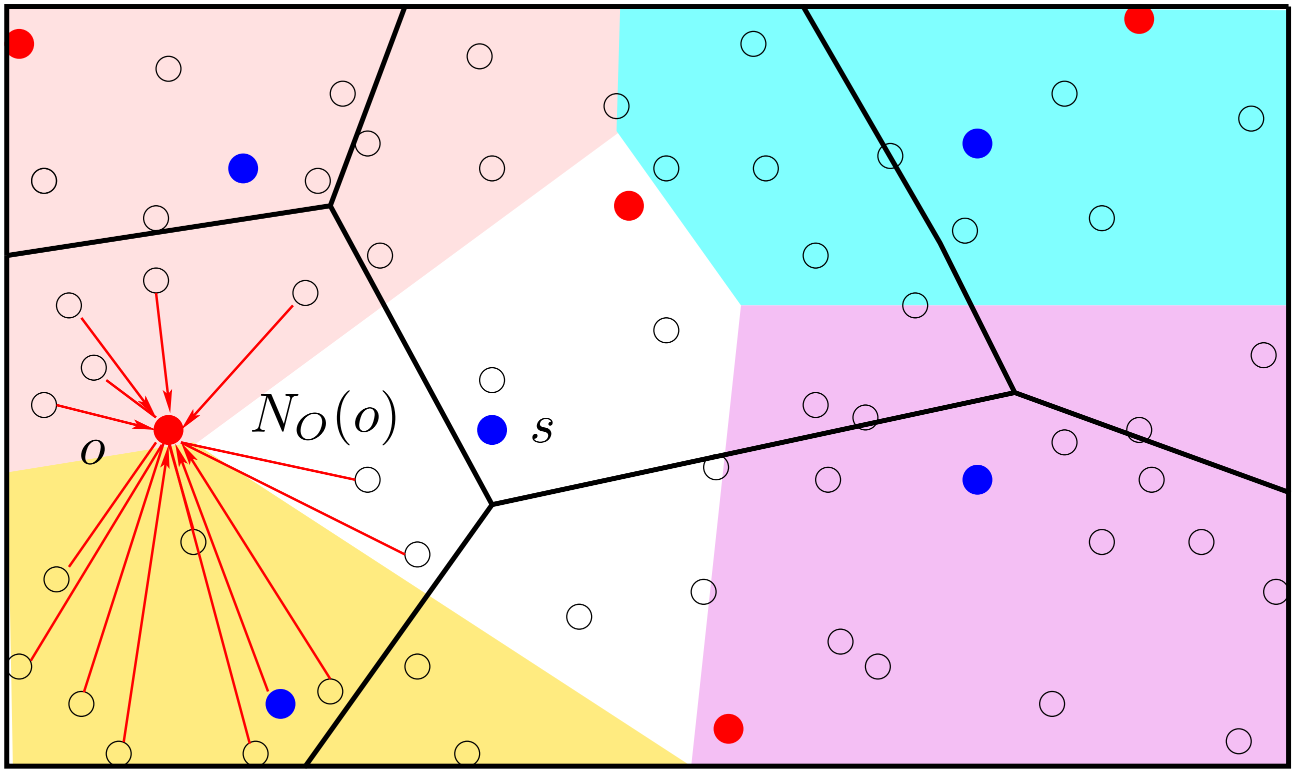
- In the solution $S - s + o$, each point is connected to the closest median in $S - s + o$.
- $cost(S - s + o)$ is the sum of distances of all the points to their nearest medians.
- We are going to demonstrate a possible way of connecting each client to a median in $S - s + o$ to get an upper bound on $cost(S - s + o)$.

Upper bound on $cost(S - s + o)$



Points in $N_O(o)$ are now connected to the new median o.

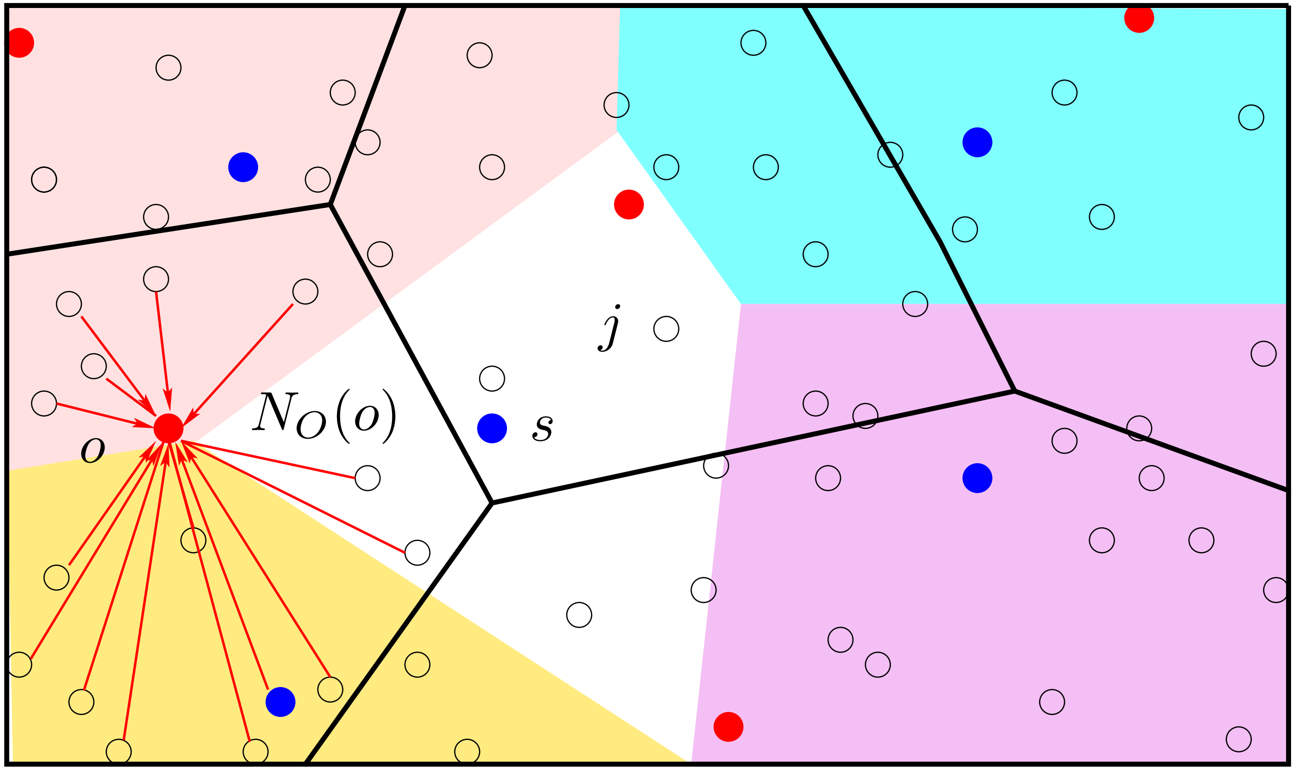
Upper bound on $cost(S - s + o)$



Thus, the increase in the distance for $j \in N_O(o)$ is at most

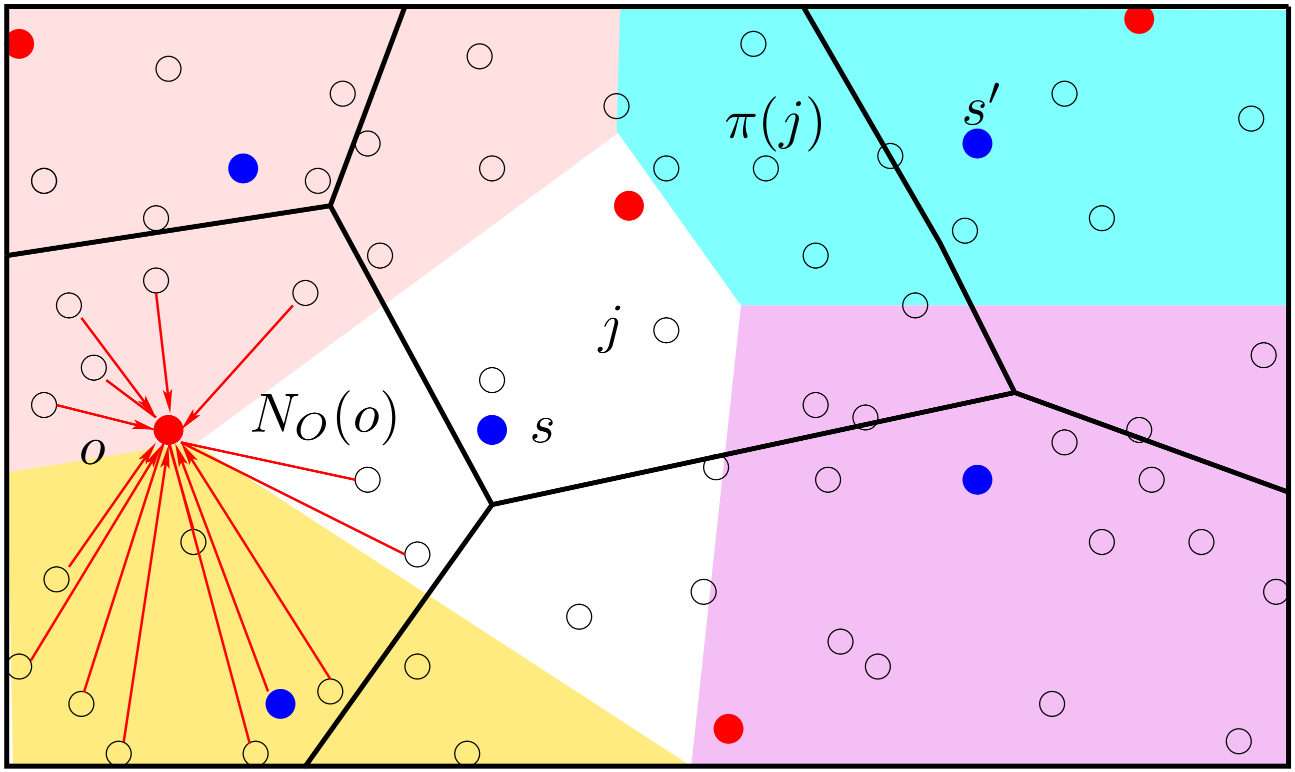
$$O_j - S_j.$$

Upper bound on $cost(S - s + o)$



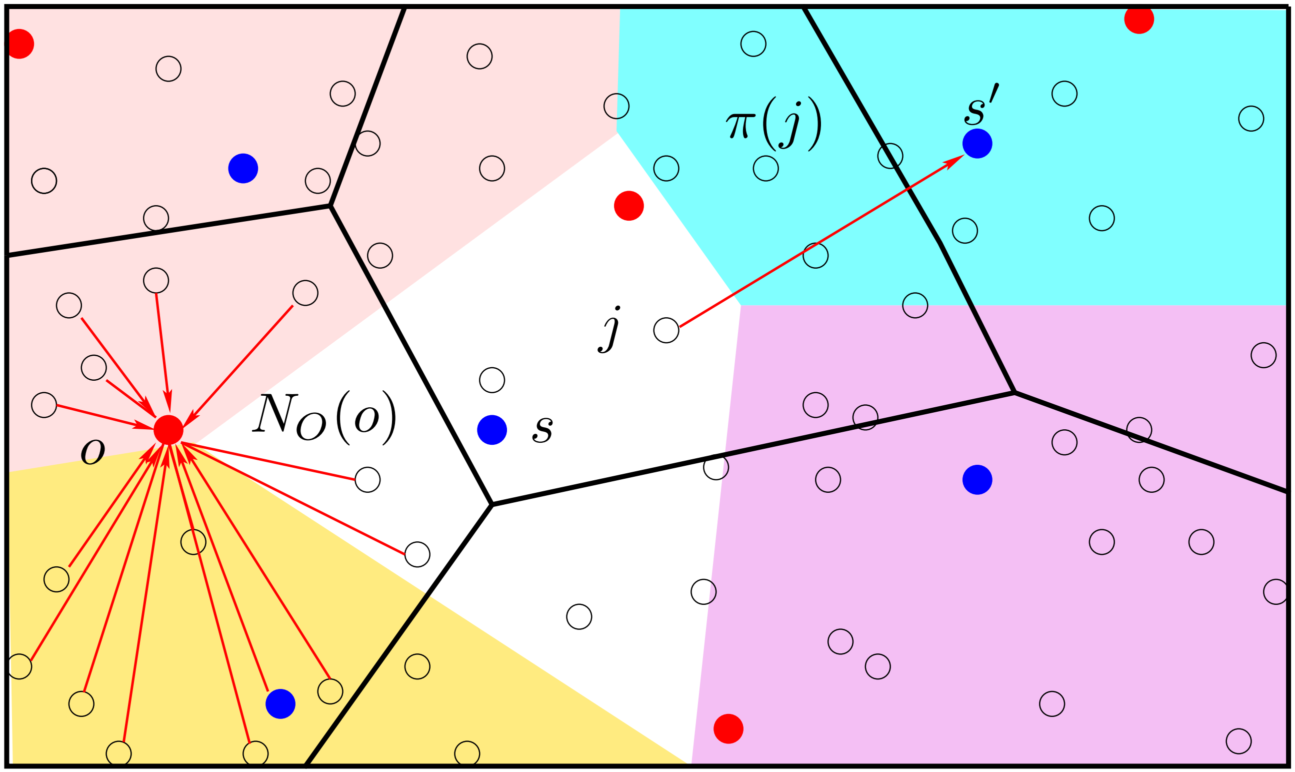
- Consider a point $j \in N_S(s) \setminus N_O(o)$.

Upper bound on $cost(S - s + o)$



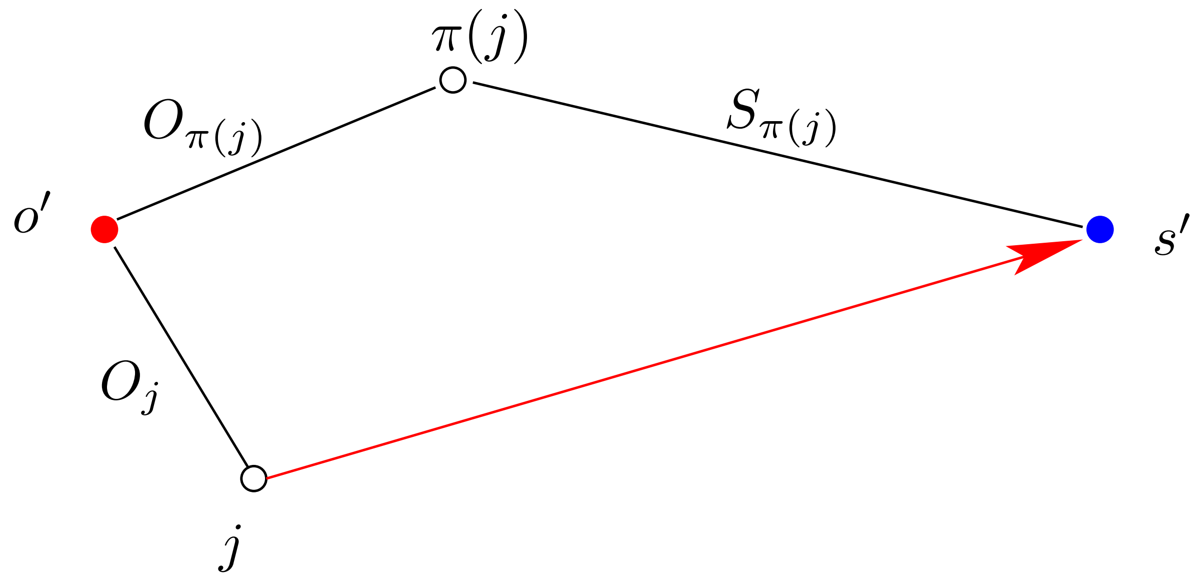
- Consider a point $j \in N_S(s) \setminus N_O(o)$.
- Suppose $\pi(j) \in N_S(s')$. (Note that $s' \neq s$.)

Upper bound on $cost(S - s + o)$



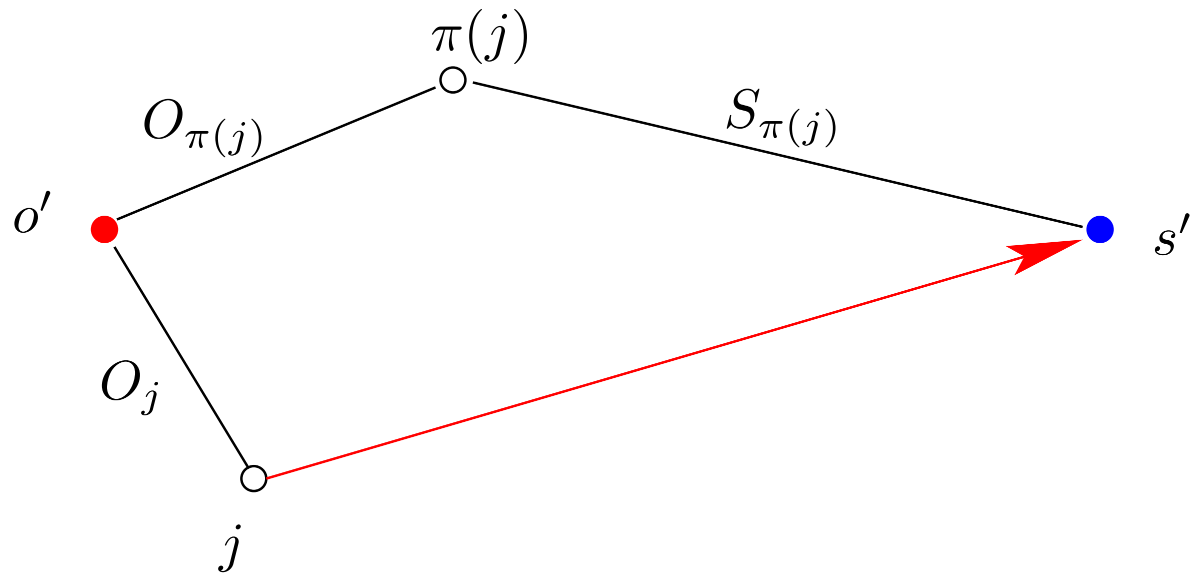
- Consider a point $j \in N_S(s) \setminus N_O(o)$.
- Suppose $\pi(j) \in N_S(s')$. (Note that $s' \neq s$.)
- Connect j to s' now.

Upper bound on $cost(S - s + o)$



- New distance of j is at most $O_j + O_{\pi(j)} + S_{\pi(j)}$.

Upper bound on $cost(S - s + o)$



- New distance of j is at most $O_j + O_{\pi(j)} + S_{\pi(j)}$.
- Therefore, the increase in the distance for $j \in N_S(s) \setminus N_O(o)$ is at most

$$O_j + O_{\pi(j)} + S_{\pi(j)} - S_j.$$

Upper bound on the increase in the cost

- Lets try to count the total increase in the cost.

Upper bound on the increase in the cost

- Lets try to count the total increase in the cost.
- Points $j \in N_O(o)$ contribute at most

$$(O_j - S_j).$$

Upper bound on the increase in the cost

- Lets try to count the total increase in the cost.
- Points $j \in N_O(o)$ contribute at most

$$(O_j - S_j).$$

- Points $j \in N_S(s) \setminus N_O(o)$ contribute at most

$$(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j).$$

Upper bound on the increase in the cost

- Lets try to count the total increase in the cost.
- Points $j \in N_O(o)$ contribute at most

$$(O_j - S_j).$$

- Points $j \in N_S(s) \setminus N_O(o)$ contribute at most

$$(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j).$$

- Thus, the total increase is at most,

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j).$$

Upper bound on the increase in the cost

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

Upper bound on the increase in the cost

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$\geq \quad cost(S - s + o) - cost(S)$$

Upper bound on the increase in the cost

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$\geq \quad cost(S - s + o) - cost(S)$$

$$\geq \quad 0$$

Plan

- We have one such inequality for each swap $\langle s, o\rangle$.

$$\sum_{j\in N_O(o)} (O_j - S_j) + \sum_{j\in N_S(s)\setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \;\geq\; 0.$$

Plan

- We have one such inequality for each swap $\langle s, o \rangle$.

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \ \geq \ 0.$$

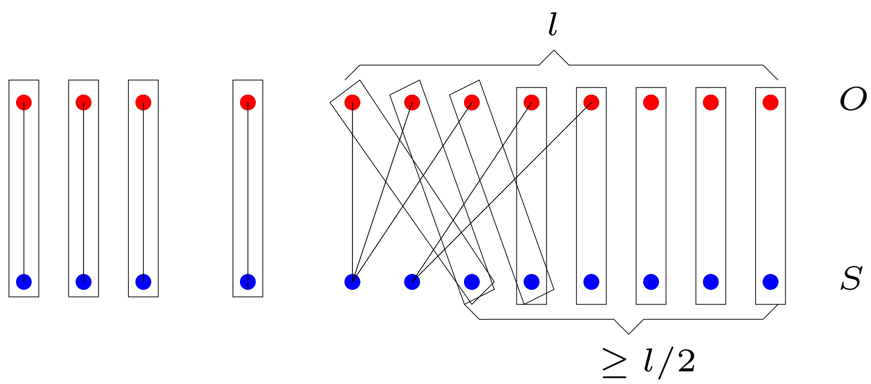
- There are k swaps that we have defined.

Plan

- We have one such inequality for each swap $\langle s, o \rangle$.

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \;\geq\; 0.$$

- There are k swaps that we have defined.

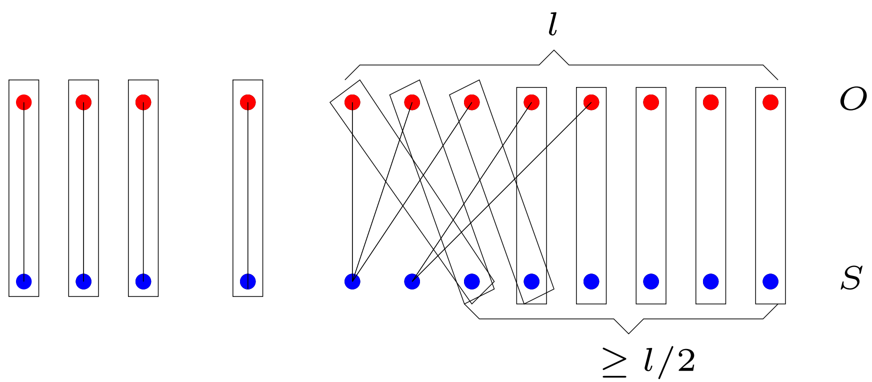


Plan

- We have one such inequality for each swap $\langle s, o \rangle$.

$$\sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \ \geq \ 0.$$

- There are k swaps that we have defined.



- Lets add the inequalities for all the k swaps and see what we get!

The first term . . .

$$\left[\sum_{j\in N_O(o)} (O_j - S_j)\right] + \sum_{j\in N_S(s)\setminus N_O(o)} \left(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j\right) \;\geq\; 0.$$

The first term ...

$$\left[\sum_{j\in N_O(o)} (O_j - S_j)\right] + \sum_{j\in N_S(s)\setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \;\geq\; 0.$$

Note that each $o \in O$ is considered in exactly one swap.

The first term ...

$$\left[\sum_{j \in N_O(o)} (O_j - S_j)\right] + \sum_{j \in N_S(s) \setminus N_O(o)} \left(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j\right) \geq 0.$$

Note that each $o \in O$ is considered in exactly one swap. Thus, the first term added over all the swaps is

$$\sum_{o \in O} \sum_{j \in N_O(o)} (O_j - S_j)$$

The first term ...

$$\left[\sum_{j \in N_O(o)} (O_j - S_j)\right] + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \;\geq\; 0.$$

Note that each $o \in O$ is considered in exactly one swap. Thus, the first term added over all the swaps is

$$\sum_{o \in O} \sum_{j \in N_O(o)} (O_j - S_j)$$

$$= \sum_j (O_j - S_j)$$

The first term ...

$$\left[\sum_{j\in N_O(o)} (O_j - S_j)\right] + \sum_{j\in N_S(s)\setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \;\geq\; 0.$$

Note that each $o \in O$ is considered in exactly one swap. Thus, the first term added over all the swaps is

$$\sum_{o\in O} \sum_{j\in N_O(o)} (O_j - S_j)$$

$$= \sum_j (O_j - S_j)$$

$$= cost(O) - cost(S).$$

The second term ...

$$\sum_{j \in N_O(o)} (O_j - S_j) + \left[\sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \right] \geq 0.$$

The second term . . .

$$\sum_{j \in N_O(o)} (O_j - S_j) + \left[\sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \right] \geq 0.$$

Note that

$$O_j + O_{\pi(j)} + S_{\pi(j)} \geq S_j.$$

The second term . . .

$$\sum_{j\in N_O(o)} (O_j - S_j) + \left[\sum_{j\in N_S(s)\setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \right] \ \geq \ 0.$$

Note that

$$O_j + O_{\pi(j)} + S_{\pi(j)} \geq S_j.$$

Thus

$$O_j + O_{\pi(j)} + S_{\pi(j)} - S_j \geq 0.$$

The second term . . .

$$\sum_{j \in N_O(o)} (O_j - S_j) + \left[\sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \right] \geq 0.$$

Note that

$$O_j + O_{\pi(j)} + S_{\pi(j)} \geq S_j.$$

Thus

$$O_j + O_{\pi(j)} + S_{\pi(j)} - S_j \geq 0.$$

Thus the second term is at most

$$\sum_{j \in N_S(s)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j).$$

The second term . . .

Note that each $s \in S$ is considered in at most two swaps.

The second term . . .

Note that each $s \in S$ is considered in at most two swaps.

Thus, the second term added over all the swaps is at most

$$2 \sum_{s \in S} \sum_{j \in N_S(s)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

The second term . . .

Note that each $s \in S$ is considered in at most two swaps.

Thus, the second term added over all the swaps is at most

$$2 \sum_{s \in S} \sum_{j \in N_S(s)} \left(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j\right)$$

$$= \; 2 \sum_{j} \left(O_j + O_{\pi(j)} + S_{\pi(j)} - S_j\right)$$

The second term ...

Note that each $s \in S$ is considered in at most two swaps.

Thus, the second term added over all the swaps is at most

$$2 \sum_{s \in S} \sum_{j \in N_S(s)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$= \; 2 \sum_{j} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$= 2 \left[\sum_j O_j + \sum_j O_{\pi(j)} + \sum_j S_{\pi(j)} - \sum_j S_j \right]$$

The second term ...

Note that each $s \in S$ is considered in at most two swaps.

Thus, the second term added over all the swaps is at most

$$2 \sum_{s \in S} \sum_{j \in N_S(s)} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$= \; 2 \sum_{j} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j)$$

$$= 2 \left[\sum_j O_j + \sum_j O_{\pi(j)} + \sum_j S_{\pi(j)} - \sum_j S_j \right]$$

$$= 4 \cdot cost(O).$$

Putting things together

$$0 \leq \sum_{\langle s,o \rangle} \left[\sum_{j \in N_O(o)} (O_j - S_j) \quad + \quad \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - \right.$$

Putting things together

$$0 \leq \sum_{\langle s,o \rangle} \left[\sum_{j \in N_O(o)} (O_j - S_j) \quad + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} - \right.$$

$$\leq [cost(O) - cost(S)] \quad + \quad [4 \cdot cost(O)]$$

Putting things together

$$0 \leq \sum_{\langle s,o \rangle} \Bigg[\sum_{j \in N_O(o)} (O_j - S_j) \quad + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} -$$

$$\leq [cost(O) - cost(S)] \quad + \quad [4 \cdot cost(O)]$$

$$= 5 \cdot cost(O) - cost(S).$$

Putting things together

$$0 \leq \sum_{\langle s,o \rangle} \left[\sum_{j \in N_O(o)} (O_j - S_j) \quad + \sum_{j \in N_S(s) \setminus N_O(o)} (O_j + O_{\pi(j)} + S_{\pi(j)} -\right.$$

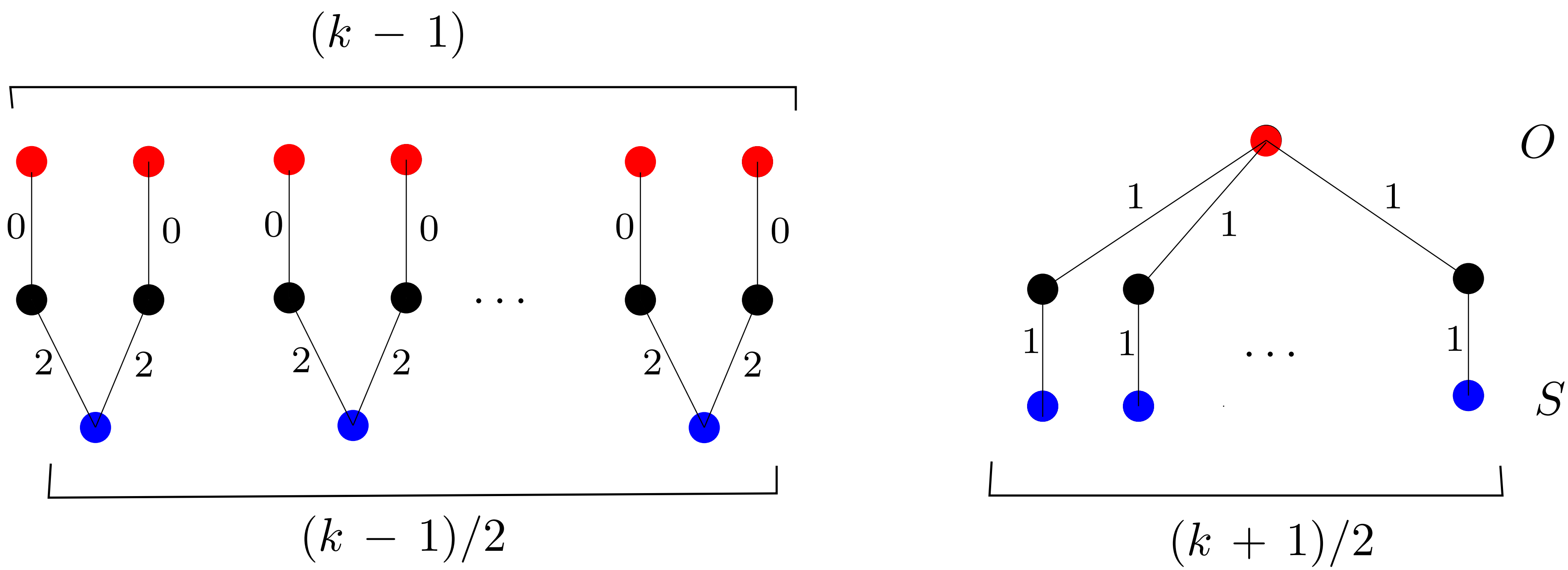
$$\leq [cost(O) - cost(S)] \quad + \quad [4 \cdot cost(O)]$$

$$= 5 \cdot cost(O) - cost(S).$$

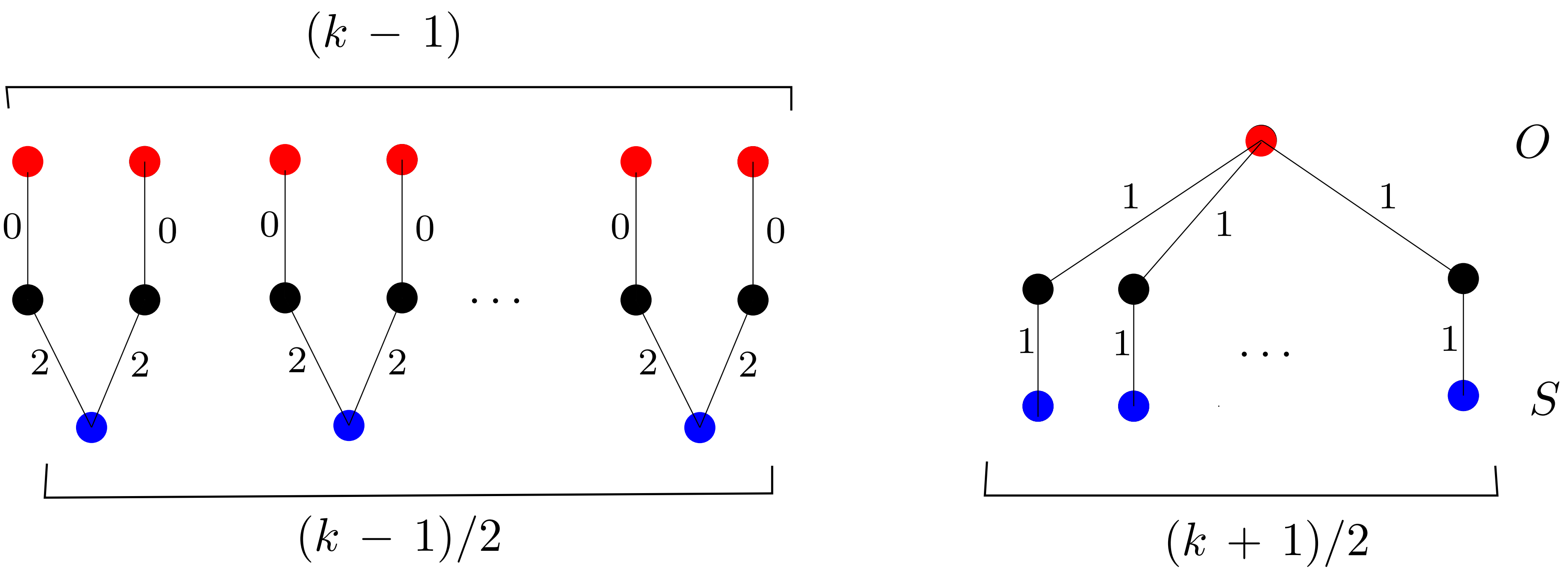
Therefore,

$$cost(S) \; \leq \; 5 \cdot cost(O).$$

A tight example

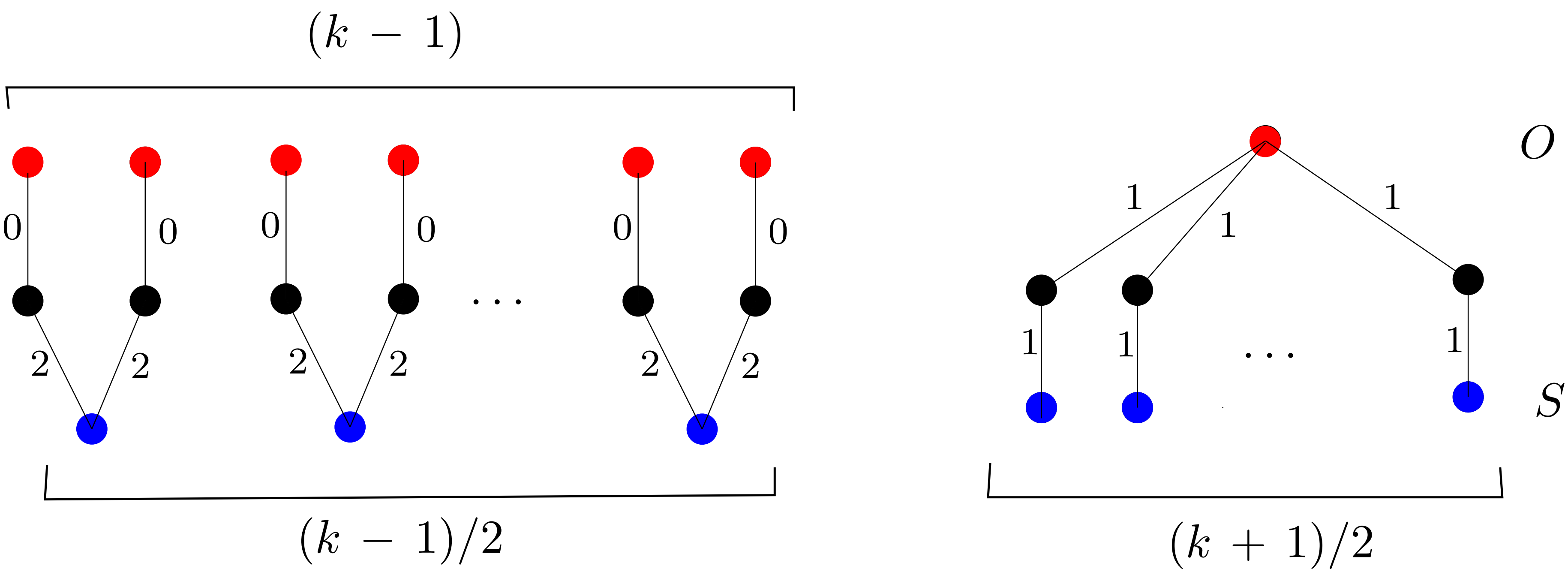


A tight example



- $cost(S) = 4 \cdot (k-1)/2 + (k+1)/2 = (5k-3)/2$

A tight example



- $cost(S) = 4 \cdot (k-1)/2 + (k+1)/2 = (5k-3)/2$
- $cost(O) = 0 + (k+1)/2 = (k+1)/2$

Future directions

- We do not have a good understanding of the structure of problems for which local search can yield approximation algorithms.
- Starting point could be an understanding of the success of local search techniques for the curious capacitated facility location (CFL) problems.
- For CFL problems, we know good local search algorithms. But, no non-trivial approximations known using other techniques like greedy, LP rounding etc.