Facility Location with Nonuniform Hard Capacities

Martin Pál   Éva Tardos   Tom Wexler
Cornell University

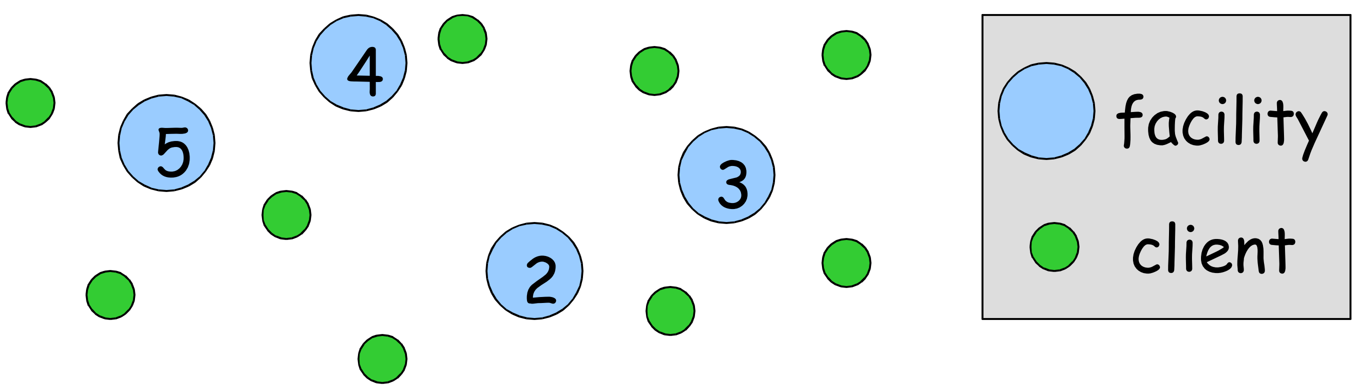
Metric Facility Location

F is a set of facilities.

D is a set of clients.

c_{ij} is the distance between any i and j in $D \cup F$.

Facility i in F has cost f_i.



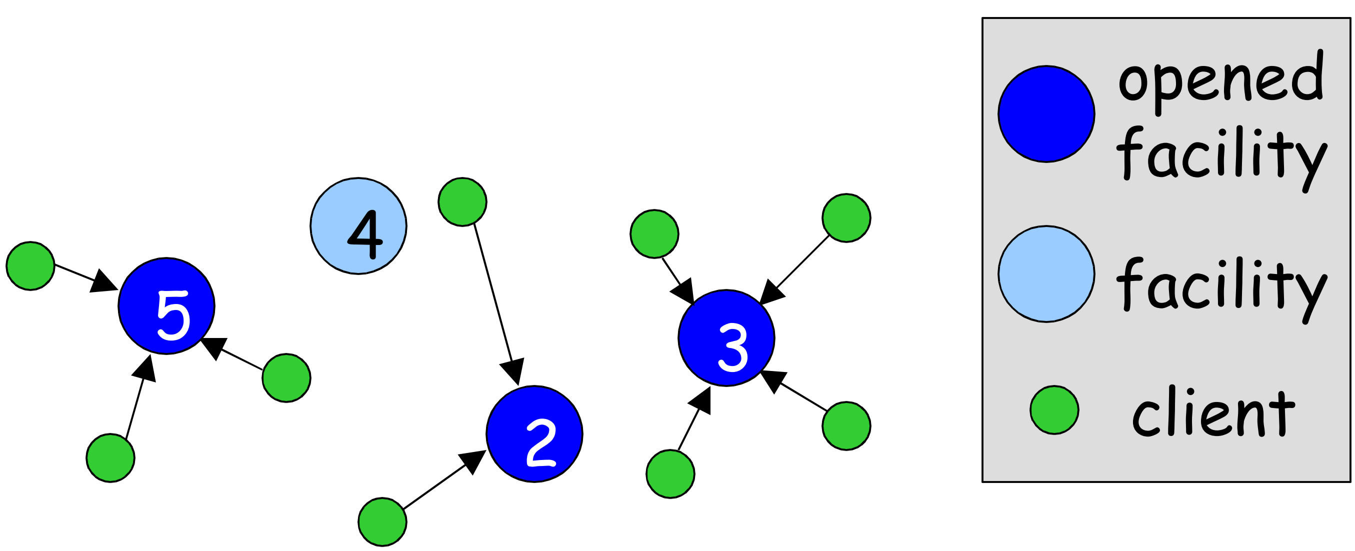
Problem Statement

We need to:

1) Pick a set S of facilities to open.

2) Assign every client to an open facility (a facility in S).

Goal: Minimize cost of S + distances.

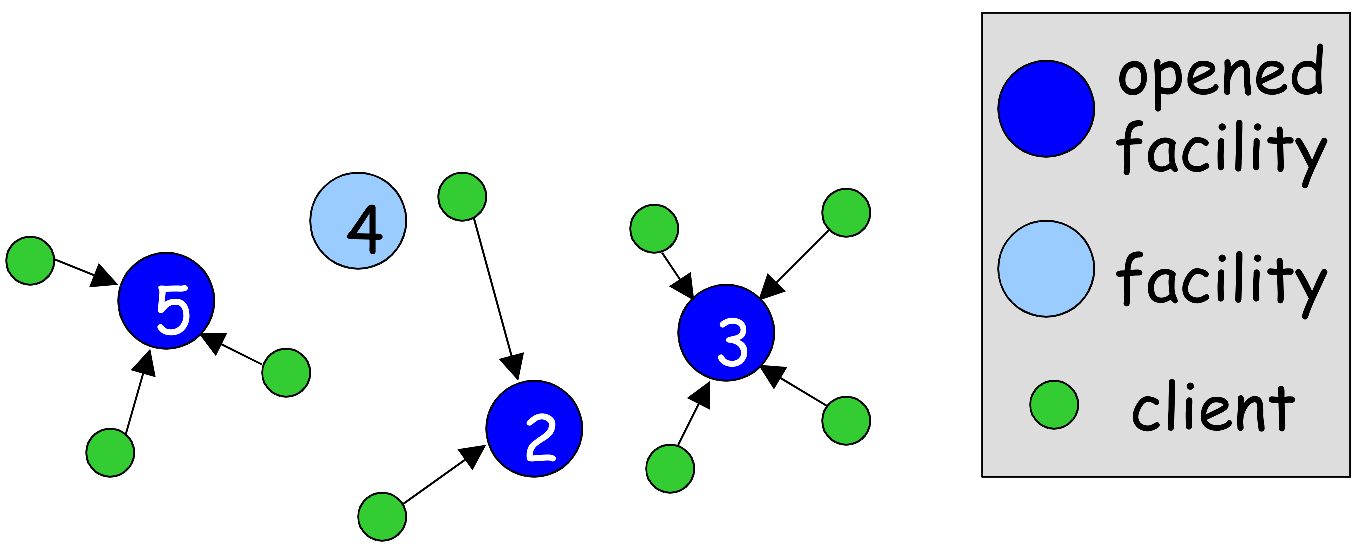


More Formally

Facility cost $c_f(S)$: sum of f_i over all i in S.

Service cost $c_s(S)$: sum of distances from clients to assigned facilities.

Goal: Minimize $c_f(S) + c_s(S)$.



Hard Capacities

Every facility i has a capacity u_i.
(how many clients i can serve)

Soft Capacities: can open k copies of facility i and serve $k \cdot u_i$ clients.

Hard Capacities: can't open multiple copies of any facility.

Note: with hard capacities, there may be no solution.

Previous Results I

Techniques Used for Fac. Loc.:

LP rounding

Primal-dual algorithms

Chudak & Shmoys '99, Jain & Vazirani '99:

LP techniques give c-approx. for soft capacities.

Problem: Known LPs have large integrality gap for hard capacities.

Previous Results II

Techniques Used for Fac. Loc.:

Local search

Korupolu et al '98, Chudak & Williamson '99:
Local search gives c-approx. for
uniform, hard capacities.

Arya et al '01:
Local search gives c-approx. for
nonuniform, soft capacities.

Our Result:

Local search gives c-approx. for
nonuniform, hard capacities.

Local Search

1) Start with any feasible solution.

2) Improve solution with
"local operations".

3) Stop when there are no remaining
operations that lower the cost.

Well, almost…

…we want to finish in poly-time:

Each operation is required to lower
cost by a factor of $1/\text{poly}(n,\varepsilon)$.

Operation 1 (of 3): add

1) add(s) – Open facility s.

If our solution opened S before, add(s) means we open $S \cup \{s\}$ now.

Given S, where do we send clients?

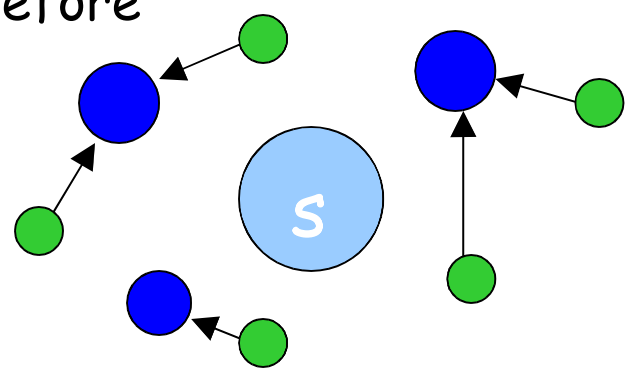
ï Without capacities, we just assign clients the the nearest open facility.

ï With capacities, we can still get optimum by solving a min cost flow.

Operation 2: open

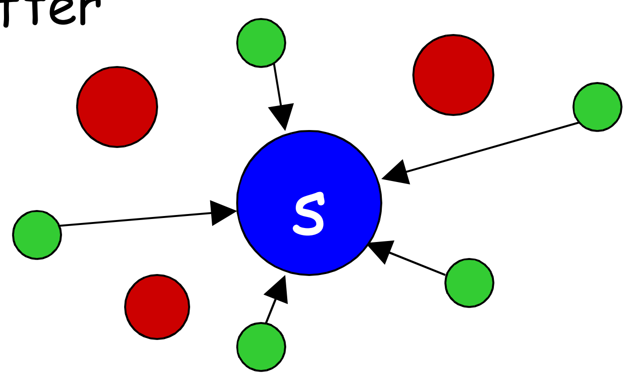
2) open(s,T) – Open facility s, send clients from T to s, and close T.

before



If we close t, we are only allowed to send t's clients to s.

after



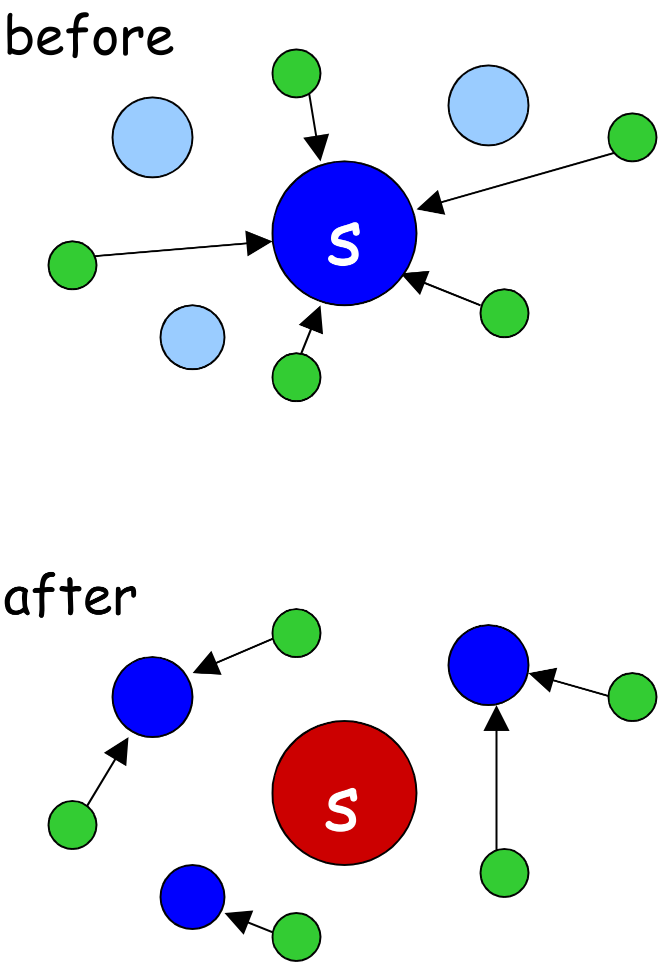
Capacity of s

≥

# of clients served by T.

Operation 3: close

3) close(s,T) – Open facilities T, send clients from s to T, close s.



Capacity of T
≥
# of clients served by s.

Can We Find Operations?

Too many operations like open(s,T) and close(s,T) to consider all!

Can we get find the best one?

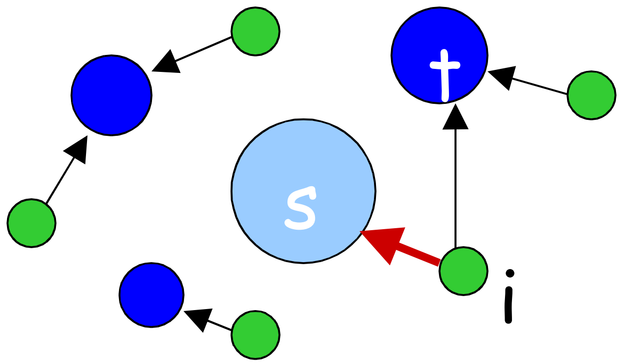
Not quite, but close enough.

Plan: For each facility s, generate a good set T for the open operation.

Likewise for close.

Good Operations

open(s, T)



Want to open s, need to pick T.

Idea: t has demand & benefit.

A knapsack problem! (have scheme)

Value(t) = f_t + reassignment costs.

Size(t) = # clients assigned to t.

$$\text{max.} \quad \sum_{t \in T} \text{Value}(t)$$

$$\text{s. t.} \quad \sum_{t \in T} \text{Size}(t) \leq \text{cap.}(s).$$

Service Cost

What can we say about the output of our algorithm?

Thm: [Korupolu et al]

If no add(s) op. improves solution S,

$$c_s(S) \leq c_s(S^{OPT}) + c_f(S^{OPT}).$$

So service cost is low. What about facility cost?

Facility Cost

Bounding $c_f(S)$:

Local opt. ➡ $0 \le$ operation costs.

(for any operation)

Every operation gives a bound:

Consider close(s, T).

$$c_f(s) \le c_f(T) + \Delta$$

benefit of closing s

cost of opening T

change in cost of shipping to T

The Plan

Idea: Pick some set of operations.

Each gives $c_f(T) \leq c_f(T') + \Delta$ for some sets T and T'.

(closed) (opened)

Pick operations so that

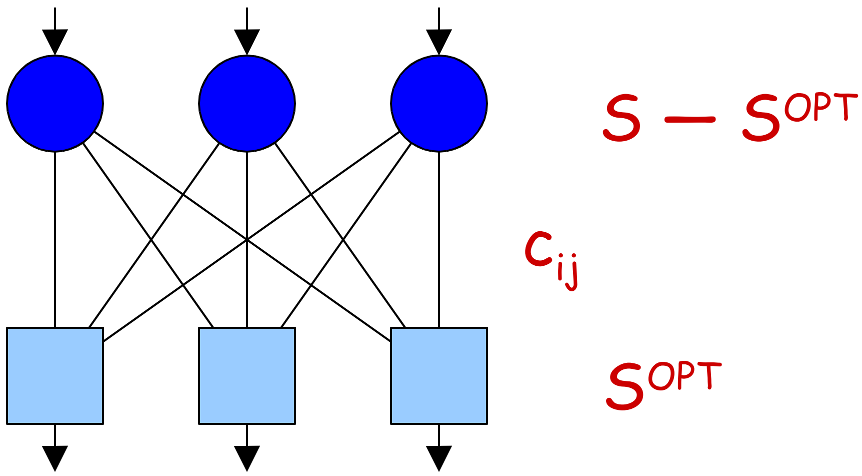
ï each s in S is closed exactly once.

ï each s in S^{OPT} is opened $\leq k$ times.

ï reassignment costs are small.

The Exchange Graph

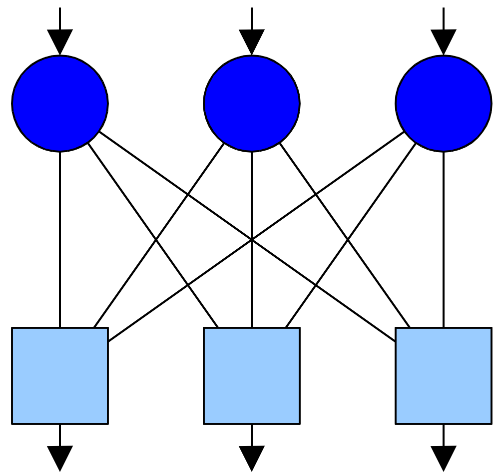
Operations should depend on S^{OPT}.



We want: swap(S^{OPT}, $S - S^{OPT}$).

We don't have such an operation.

Looking For Operations



y = flow of clients from S to S^{OPT}.

$cost(y) \le c_s(S) + c_s(S^{OPT})$

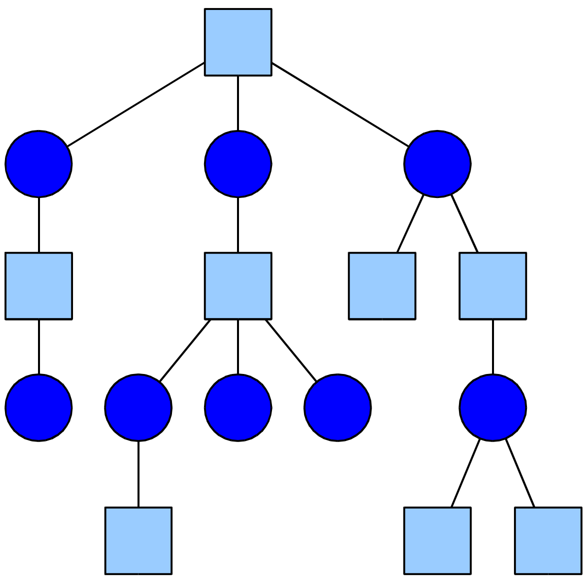
Cost of $swap(S^{OPT}, S - S^{OPT}) \le$

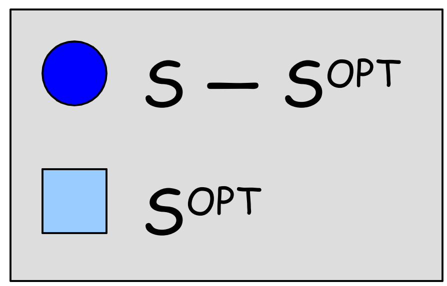
$cost(y) + c_f(S^{OPT}) - c_f(S)$.

What is the Flow?

Conveniently, the flow is a tree!

(or rather a forest... just augment cycles)





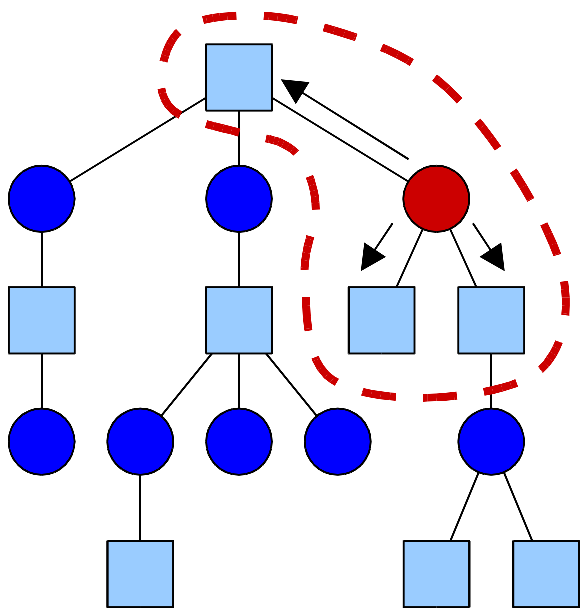
How do we get operations from this?

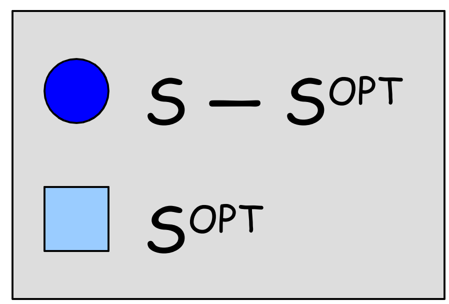
Remember, we will use inqualities like $c_f(T) \leq c_f(T') + \Delta$.

Since flow is cheap, Δ is small.

First Attempt

Just use close(s,T), picking T to be the neighbors of s in the tree.

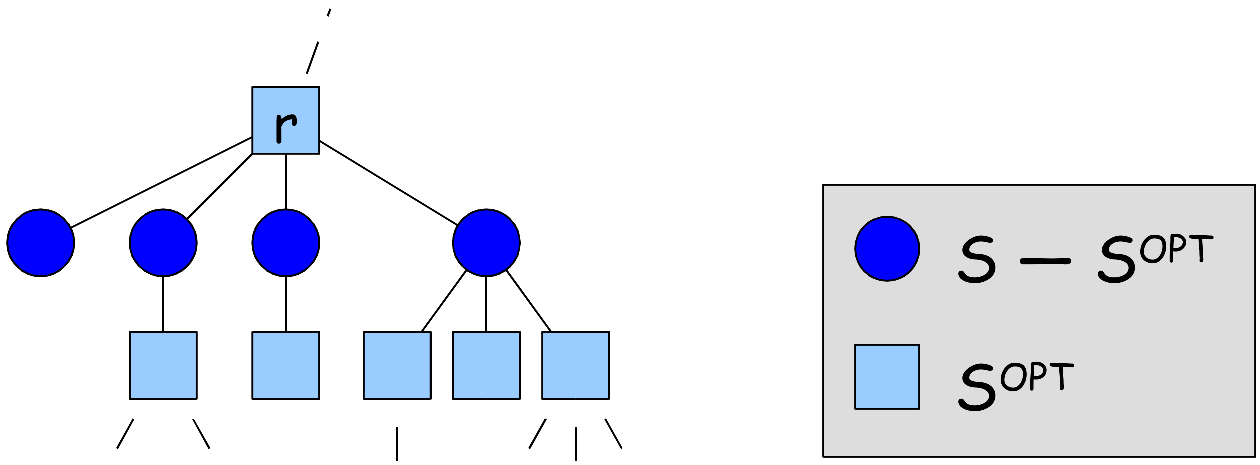


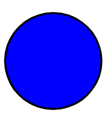


Problem: Expensive facilities in S^{OPT} may be opened many times.

Instead…

Consider subtrees of depth 2.



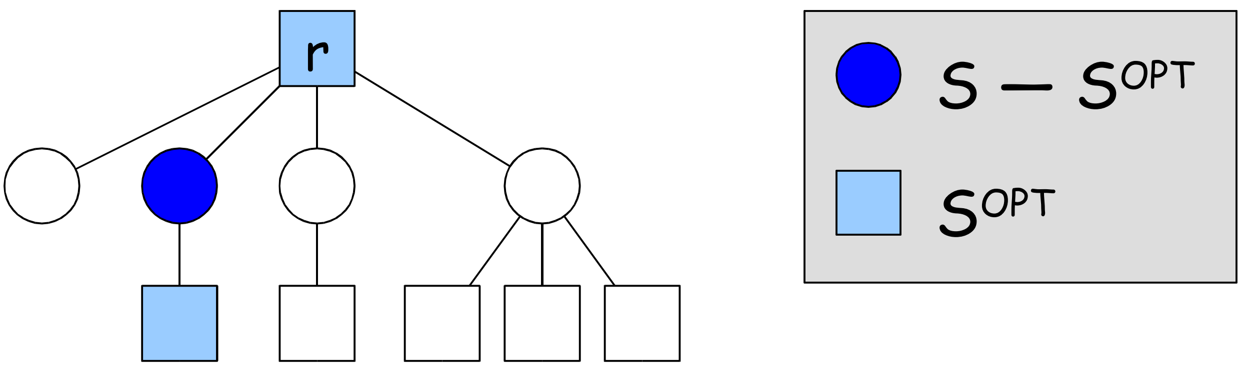
Each  node needs to be closed.

Partition these nodes into 3 sets.
We'll handle each set separately.

Heavy Nodes

Heavy Nodes: nodes responsible for > ½ the clients sent to r

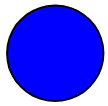
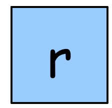
Note: There can be only one.

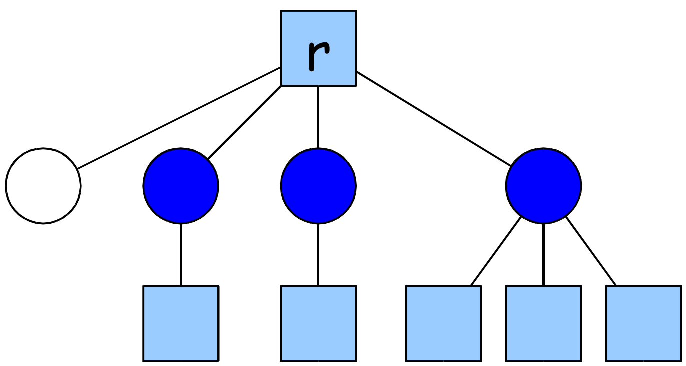


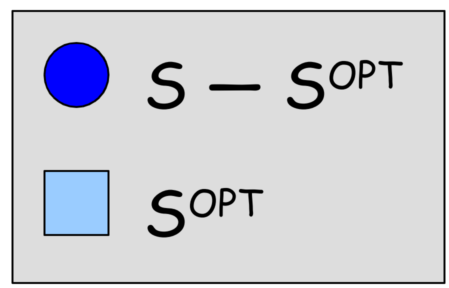
Only 1, so close it & open neighbors.

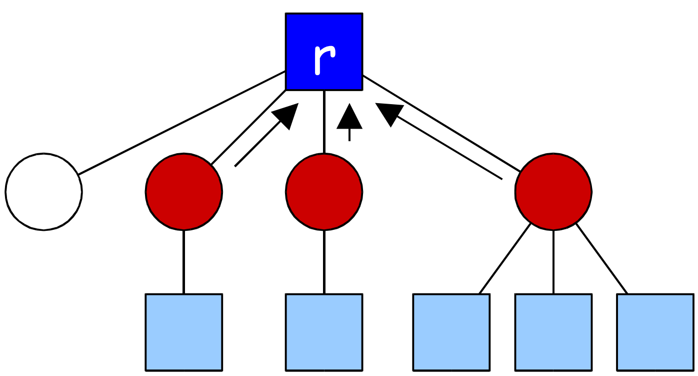
(i.e. use the naïve approach)

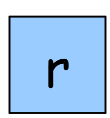
Light Dominating

Light Dominating:  nodes that send ≥ ½ their clients up to 



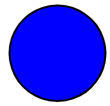


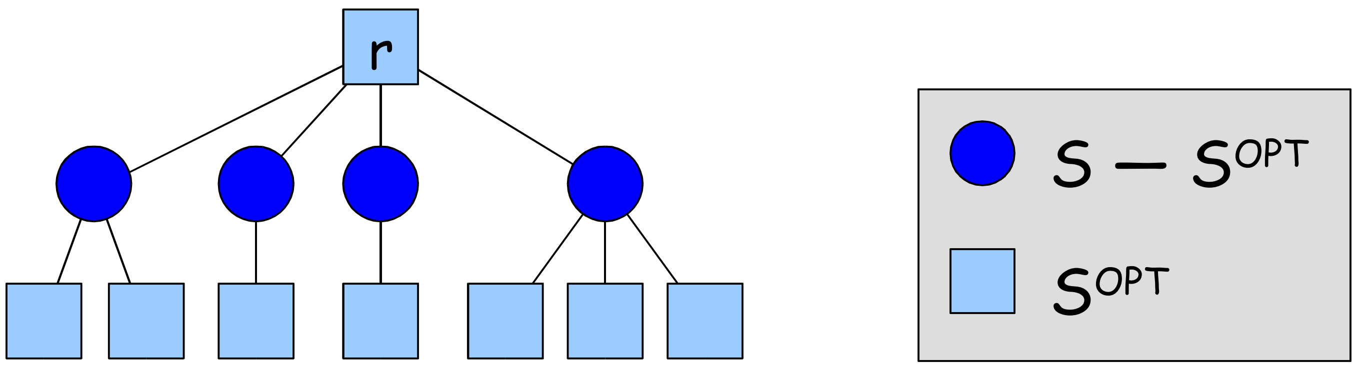


Worst case,this doubles 

's capacity: Split into a few operations.

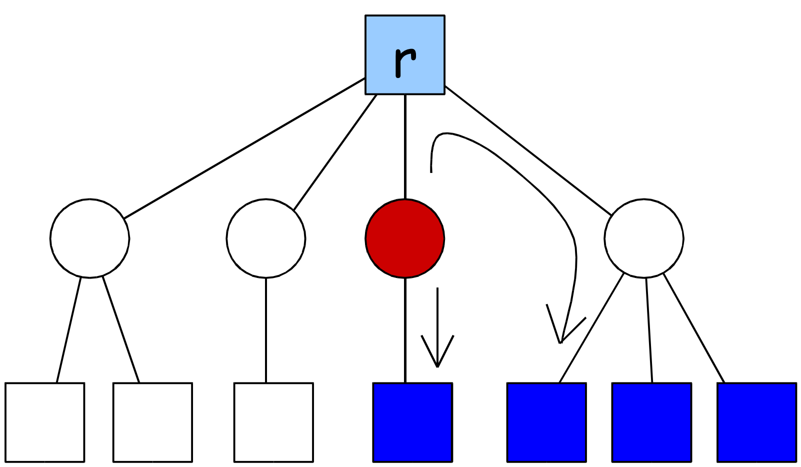
Last Case

Light Nondominating:  nodes that send ≥ ½ their clients down.

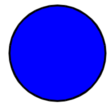


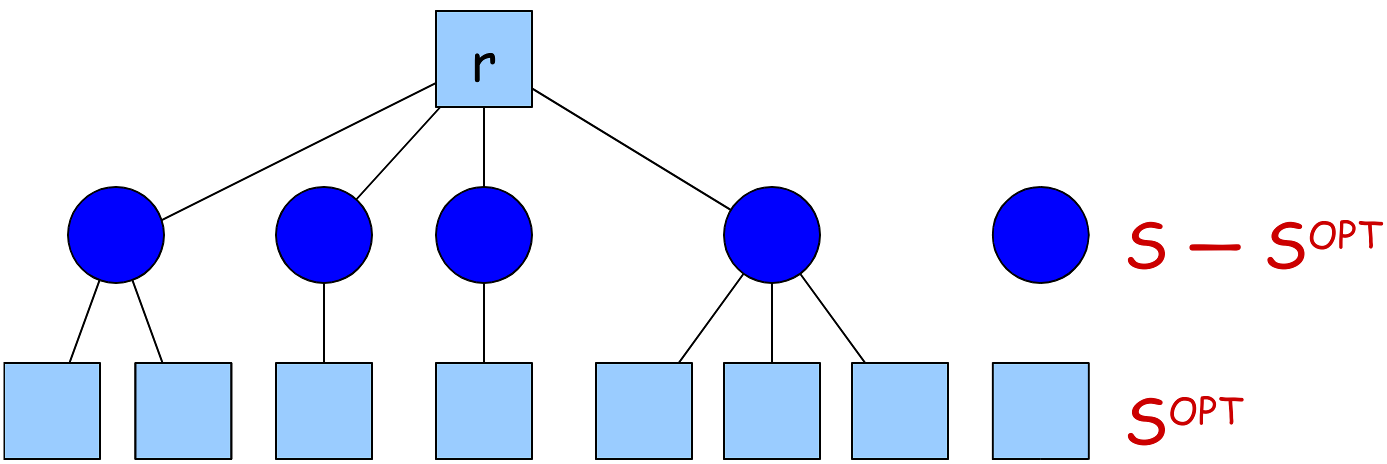
Order these by # of clients sent up.

To close facility i open: C(i), C(i+1)



Last Case

Light Nondominating:  nodes that send ≥ ½ their clients down.



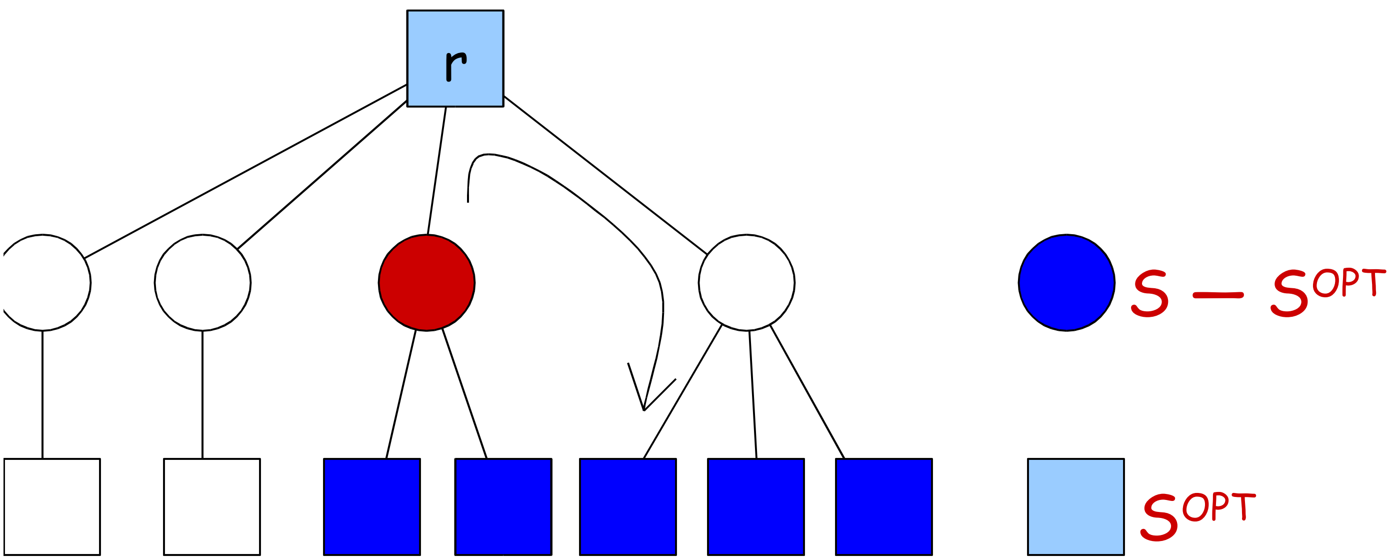
Order these by # of clients sent up.

To close facility i in LN, open:

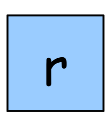
The children of i

The children of i+1

Last Case (cont')



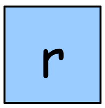
For the last LN node, we open the root instead.

For LN in total, 

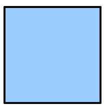
is opened once, and the bottom facilities are opened at most twice.

In Total

For each subtree,



is opened at most 4 times,


s are opened at most 2 times.

Every facility in S^{OPT} is a root of 1 subtree and a child in 1 subtree.

Altogether

ï Every node in S is closed

ï Every node is S^{OPT} is opened at most 6 times.

Result

Thm: Our local search algorithm gives a 9-approx. for facility location with hard, nonuniform capacities.

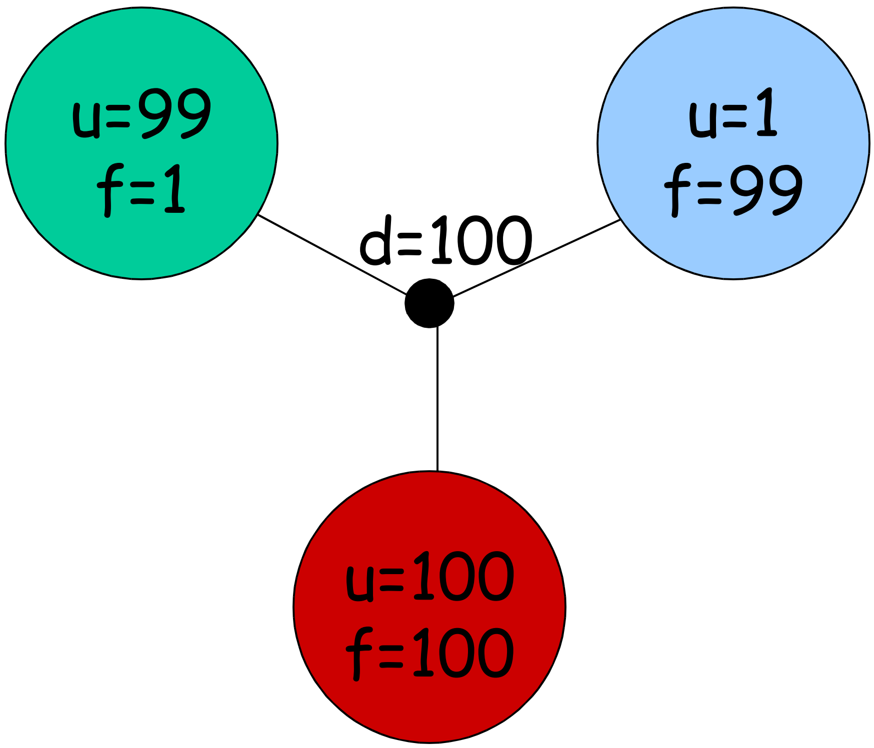
Note 1: Actually a $(9 + \epsilon)$-approx.

Note 2: Scaling lowers it to 8.53…

Note 3: At best, algorithm is a 4-approx.

LP Integrality Gap

Why not use LPs for this problem?



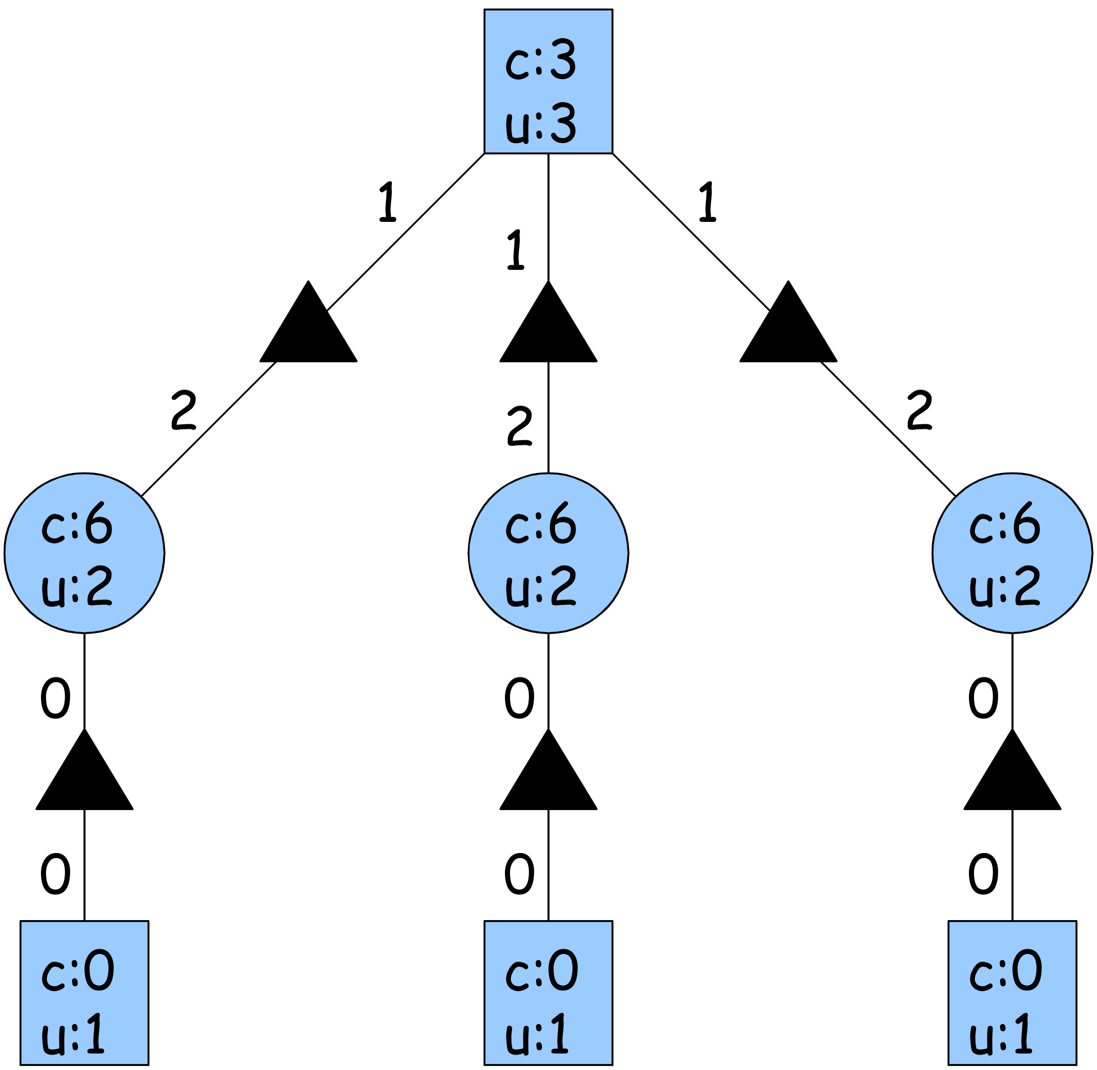
The LP solution fully opens the green facility and opens 1/100th of the red facility, thus paying a facility cost of 2.

Any integer solution pays 100.

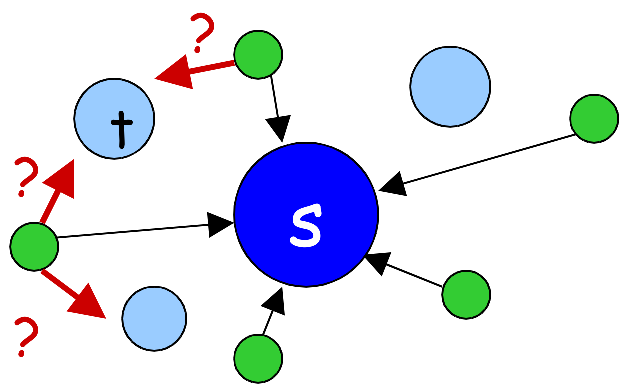
Lower Bound

How much better might our alg. be?

Thm: Alg. is at best a 4-approx.



The close(s, T) Operation



Knapsack? value(t) isn't well defined:
Which clients should t serve?

Solution: Use Δ ≠ to get an upper estimate for rerouting cost.

To send demand to t, first ship it to s, then to t.

Now value(t) = cap.(t) * dist.(s, t)
size(t) = cap.(t)

(now we have a covering knapsack problem)