Efficient **Parallel** approximation algorithms

lessons from facility location

Kanat Tangwongsan
Carnegie Mellon University

One thing I learned from greedy facility location....

which you can apply to set cover, max cover, min-sum set cover, etc.

(disclaimer: presented mostly without proof and might contain shameless plugs for upcoming talks)

Design Guidelines & Model

How to get the most out of our cores?

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

only necessary work

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

only necessary work

good locality

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

only necessary work

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

depth - longest chain of dependencies

“small”

polylog depth

only necessary work

Design Guidelines & Model

How to get the most out of our cores?

lots of parallelism

depth - longest chain of dependencies

“small”

polylog depth

only necessary work

work - total operation count

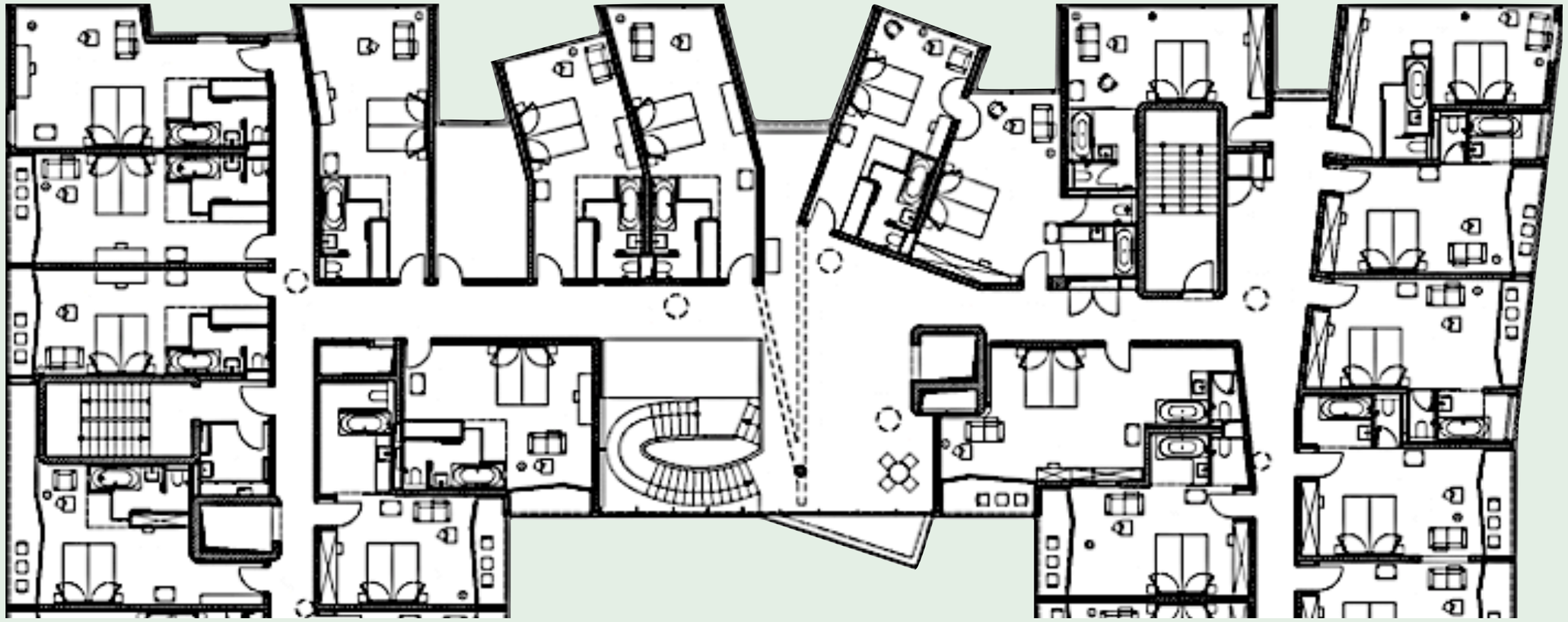
same as the sequential counterpart

Metric Facility Location

Free Food Project at Carnegie Mellon

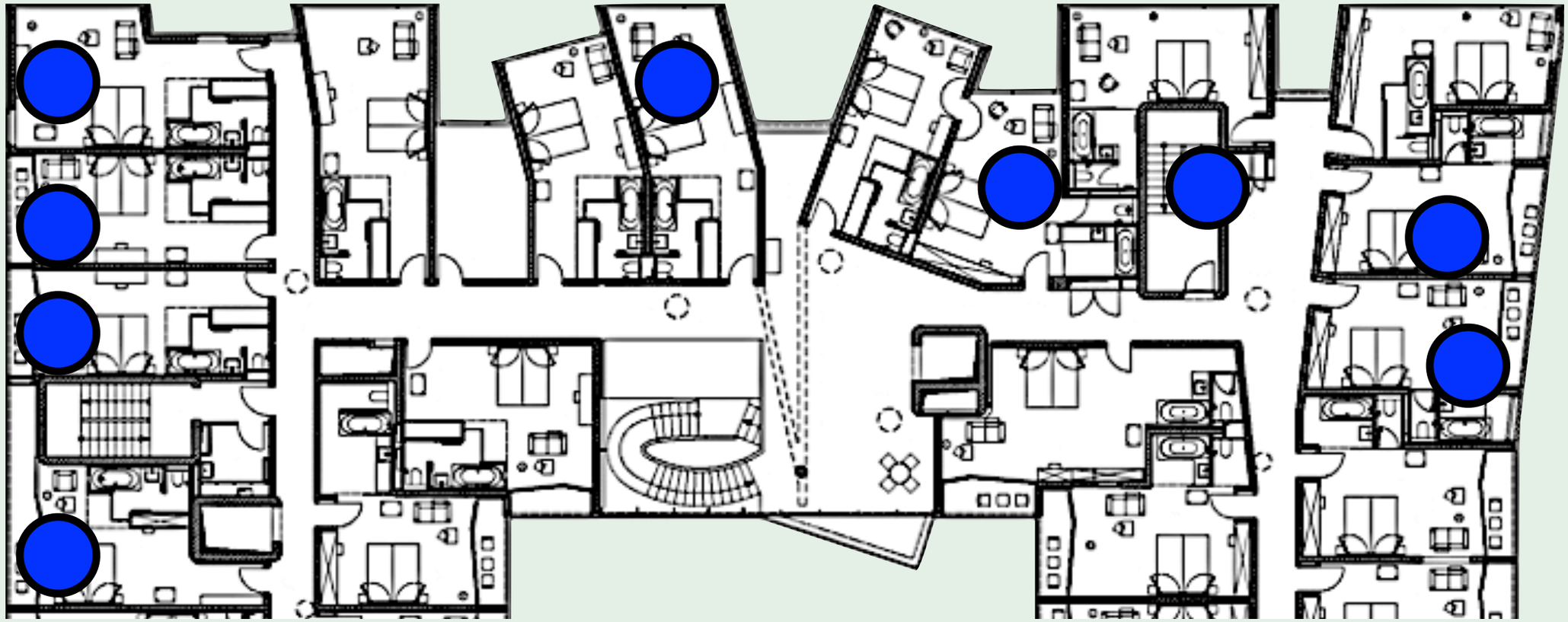
Metric Facility Location

Free Food Project at Carnegie Mellon



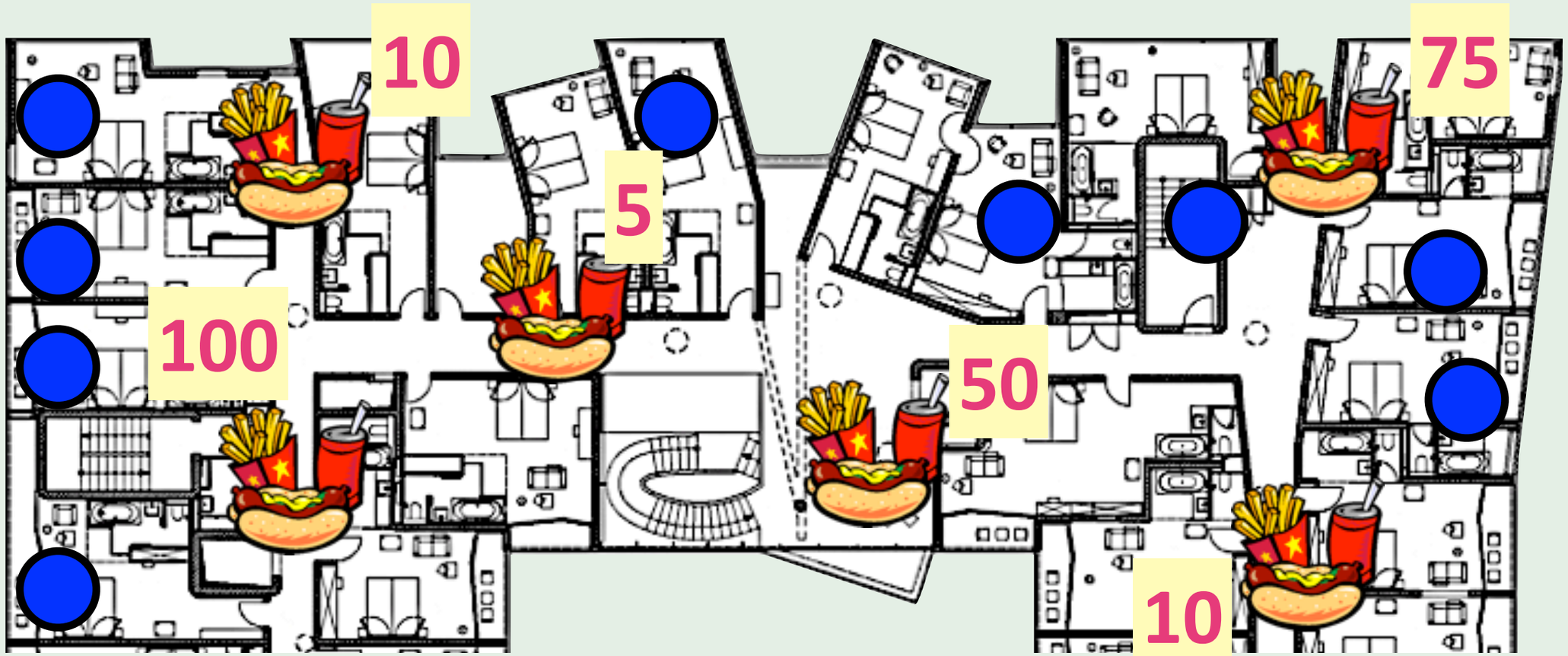
Metric Facility Location

Free Food Project at Carnegie Mellon



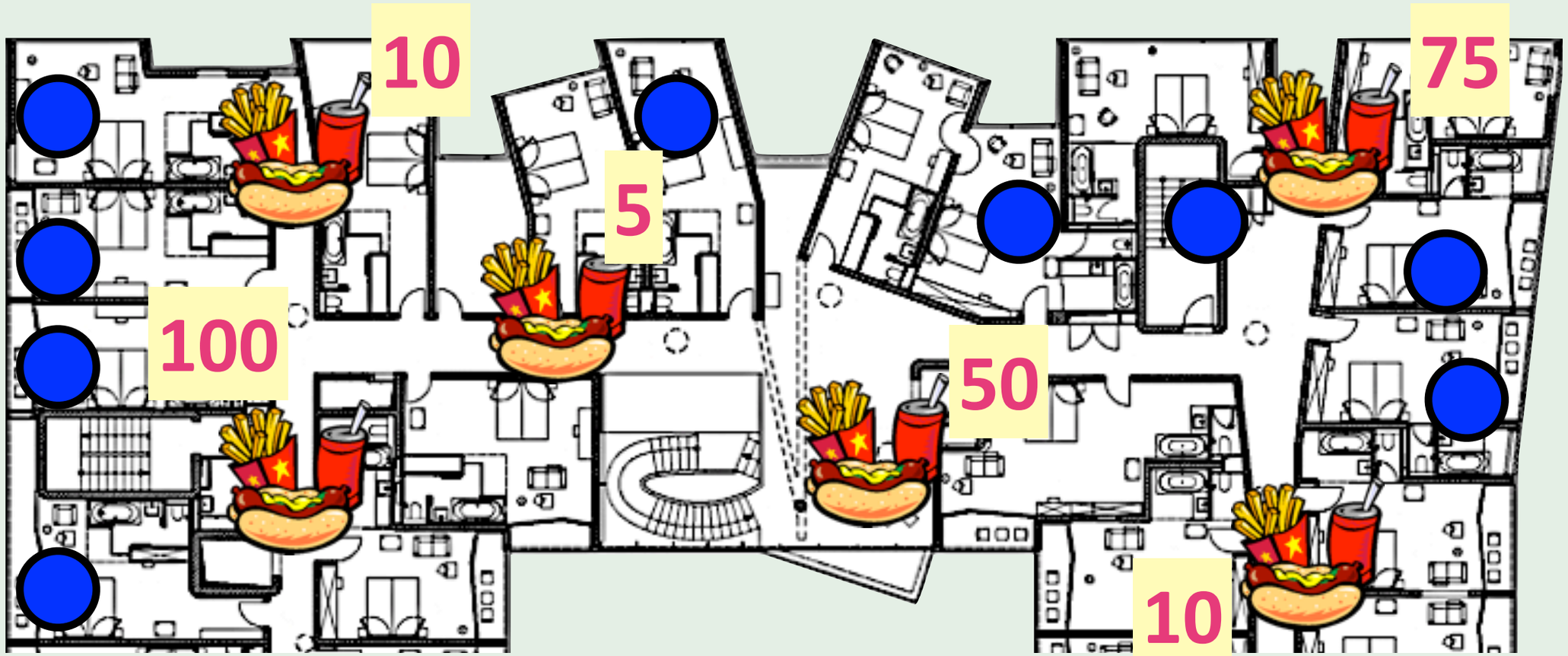
Metric Facility Location

Free Food Project at Carnegie Mellon



Metric Facility Location

Free Food Project at Carnegie Mellon



Q: Which food stations to open to minimize the cost?

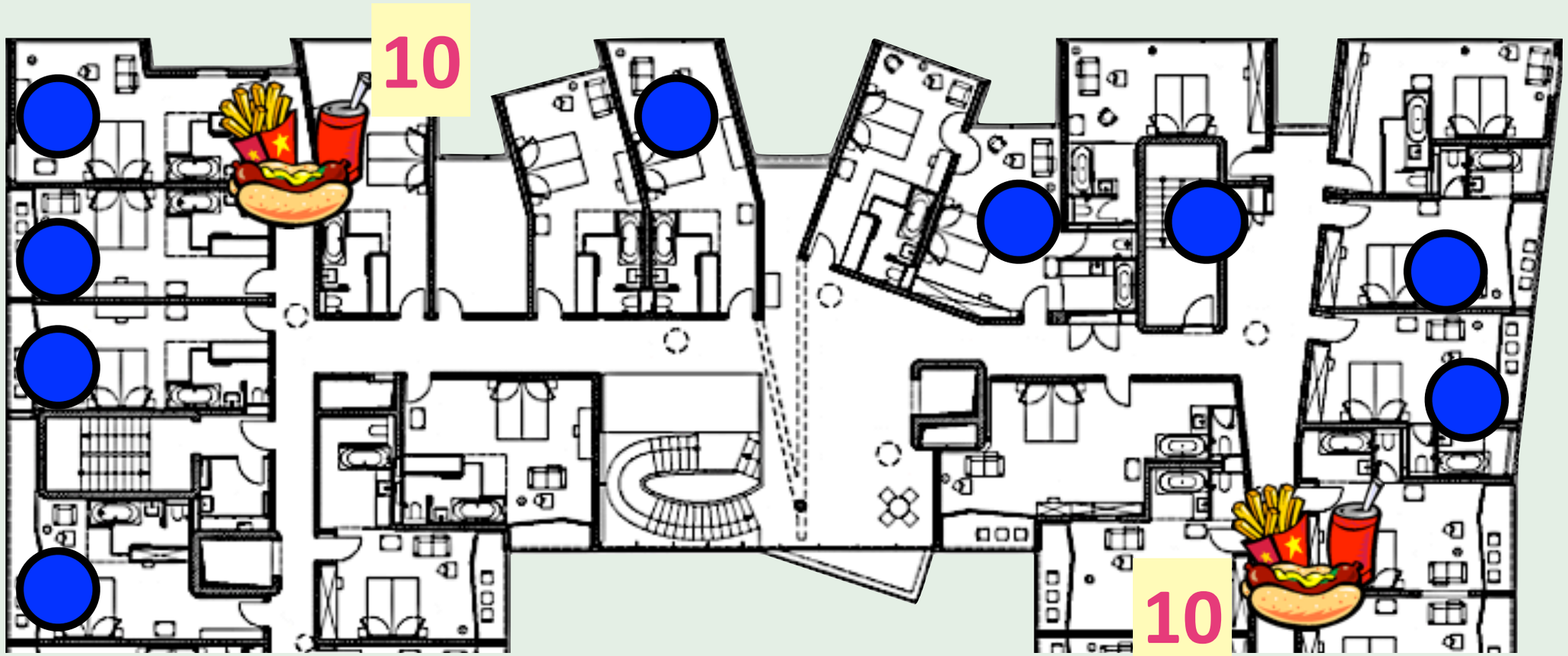
Cost = Facility Cost + Connection Cost

$$\sum_{i:\text{opened}} f_i$$

$$\sum_{j \in C} d(j, \text{closest opened})$$

Metric Facility Location

Free Food Project at Carnegie Mellon



Q: Which food stations to open to minimize the cost?

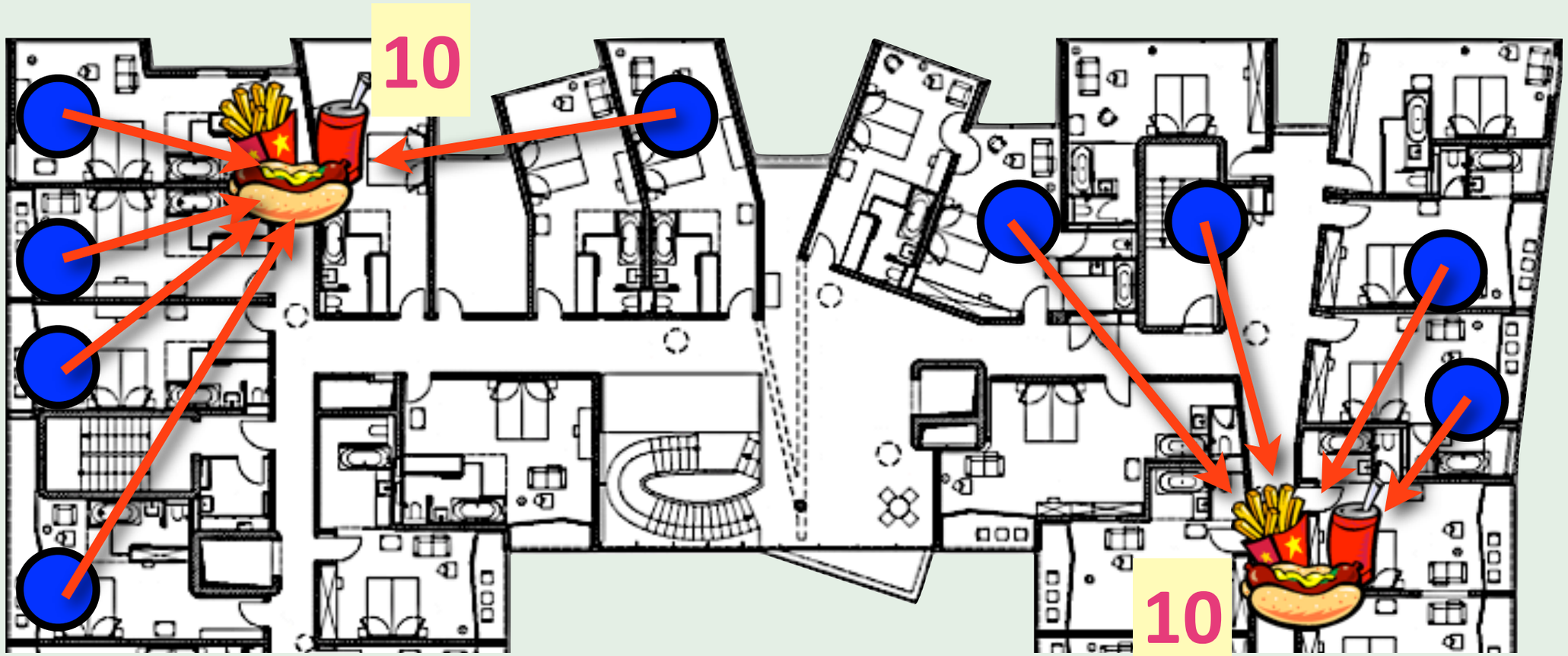
Cost = Facility Cost + Connection Cost

$$\sum_{i:\text{opened}} f_i$$

$$\sum_{j \in C} d(j, \text{closest opened})$$

Metric Facility Location

Free Food Project at Carnegie Mellon



Q: Which food stations to open to minimize the cost?

Cost = Facility Cost + Connection Cost

$$\sum_{i:\text{opened}} f_i$$

$$\sum_{j \in C} d(j, \text{closest opened})$$

Metric Facility Location

Formal Definition

Input: facilities $\boldsymbol{F}$; cost f_i

clients $\boldsymbol{C}$

connection cost $d(i, j)$

Metric Facility Location

Formal Definition

obeys triangle inequality

Input: facilities $\boldsymbol{F}$; cost f_i

clients $\boldsymbol{C}$

connection cost $d(i, j)$

Metric Facility Location

Formal Definition

obeys triangle inequality

Input: facilities ***F***; cost f_i

clients ***C***

connection cost $d(i, j)$

Task: choose facilities $F_A \subseteq \boldsymbol{F}$ to minimize

$$\textbf{Cost} = \sum_{i \in F_A} f_i + \sum_{j \in C} d(j, F_A)$$

Facility Cost + **Connection Cost**

Metric Facility Location

Formal Definition

obeys triangle inequality

Input: facilities ***F***; cost f_i

clients ***C***

connection cost $d(i, j)$

Applications:

clustering,
network design,
"testbed" for techniques, ...

Task: choose facilities $F_A \subseteq \boldsymbol{F}$ to minimize

$$\textbf{Cost} = \sum_{i \in F_A} f_i + \sum_{j \in C} d(j, F_A)$$

Facility Cost + **Connection Cost**

Metric Facility Location

What was known about this problem?

- **Hardness**

  NP-hard and factor-1.463 is hard [Guha and Khuller '99]

- **Several Constant Approximation Algorithms**

factor-4	linear program (LP) rounding	[Shmoys et al.'97]
factor-3	primal dual	[Jain and Vazirani'01]
factor-1.861	greedy	[Jain et al.'03]
	...	
factor-1.5	LP + scaling + ...	[Byrka'07]

Metric Facility Location

What was known about this problem?

- **Hardness**

  NP-hard and factor-1.463 is hard [Guha and Khuller '99]

- **Several Constant Approximation Algorithms**

factor-4	linear program (LP) rounding	[Shmoys et al.'97]
factor-3	primal dual	[Jain and Vazirani'01]
factor-1.861	greedy	[Jain et al.'03]
	...	
factor-1.5	LP + scaling + ...	[Byrka'07]

Metric Facility Location

What was known about this problem?

- **Hardness**

  NP-hard and factor-1.463 is hard [Guha and Khuller '99]

- **Several Constant Approximation Algorithms**

  factor-4 linear program (LP) rounding [Shmoys et al.'97]

  factor-3 primal dual [Jain and Vazirani'01]

Theorem:

RNC O(*m* log *m*)-work, factor-(1.861+ε) *greedy-style* approximation algorithm.

  factor-1.5 LP + scaling + … [Byrka'07]

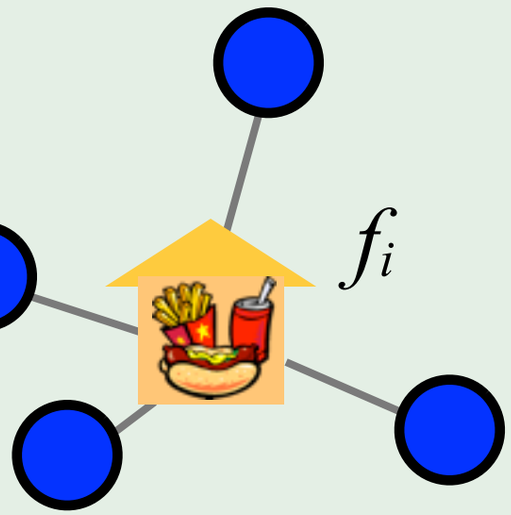
Greedy Facility Location

Jain et. al's Sequential Algorithm

Greedy Facility Location

Jain et. al's Sequential Algorithm

Star:



facility clients

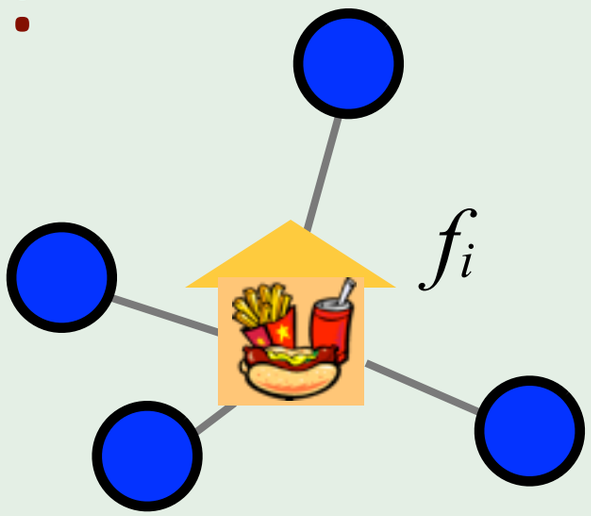
$$\mathcal{S} = (i, T \subseteq C)$$

$$\textbf{price}(S) = \frac{f_i \ + \ \sum_{j \in T} d(j,i)}{\#\text{clients in } \ T}$$

Greedy Facility Location

Jain et. al's Sequential Algorithm

Star:



facility clients

$$\mathcal{S} = (i, T \subseteq C)$$

$$\textbf{price}(S) = \frac{f_i \ + \ \sum_{j \in T} d(j,i)}{\#\text{clients in } \ T}$$

Algorithm: **factor-1.861 approx**

While (C not empty)

1. Each facility i finds the cheapest star centered at i

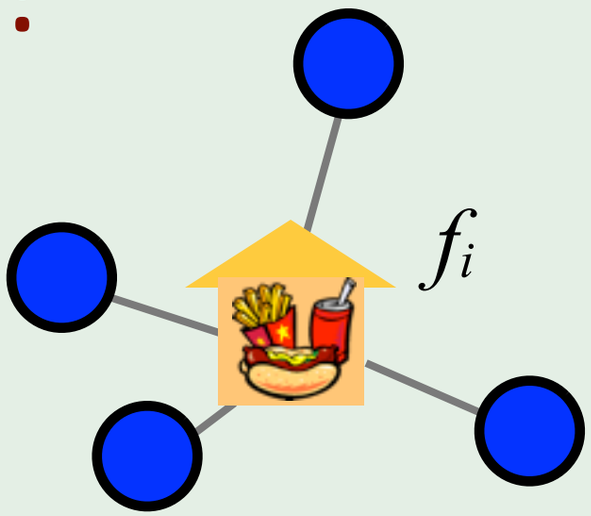
2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

Greedy Facility Location

Jain et. al's Sequential Algorithm

Star:



facility clients

$$\mathcal{S} = (i, T \subseteq C)$$

$$\textbf{price}(S) = \frac{f_i \ + \ \sum_{j \in T} d(j,i)}{\#\text{clients in } \ T}$$

Algorithm: **factor-1.861 approx**

While (C not empty) parallelizable: prefix sum

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

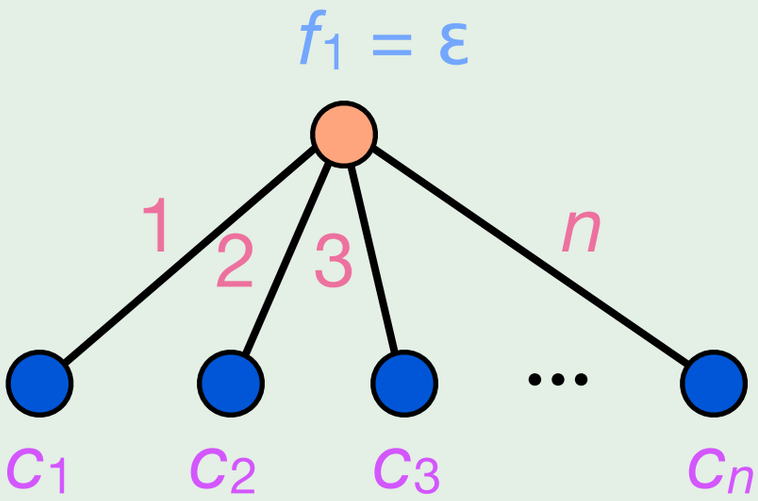
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

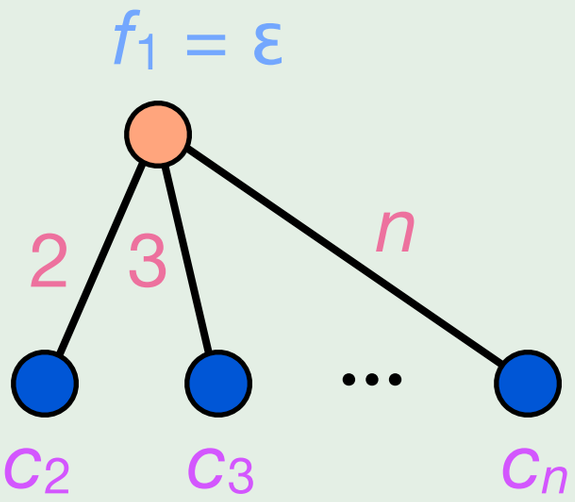
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:

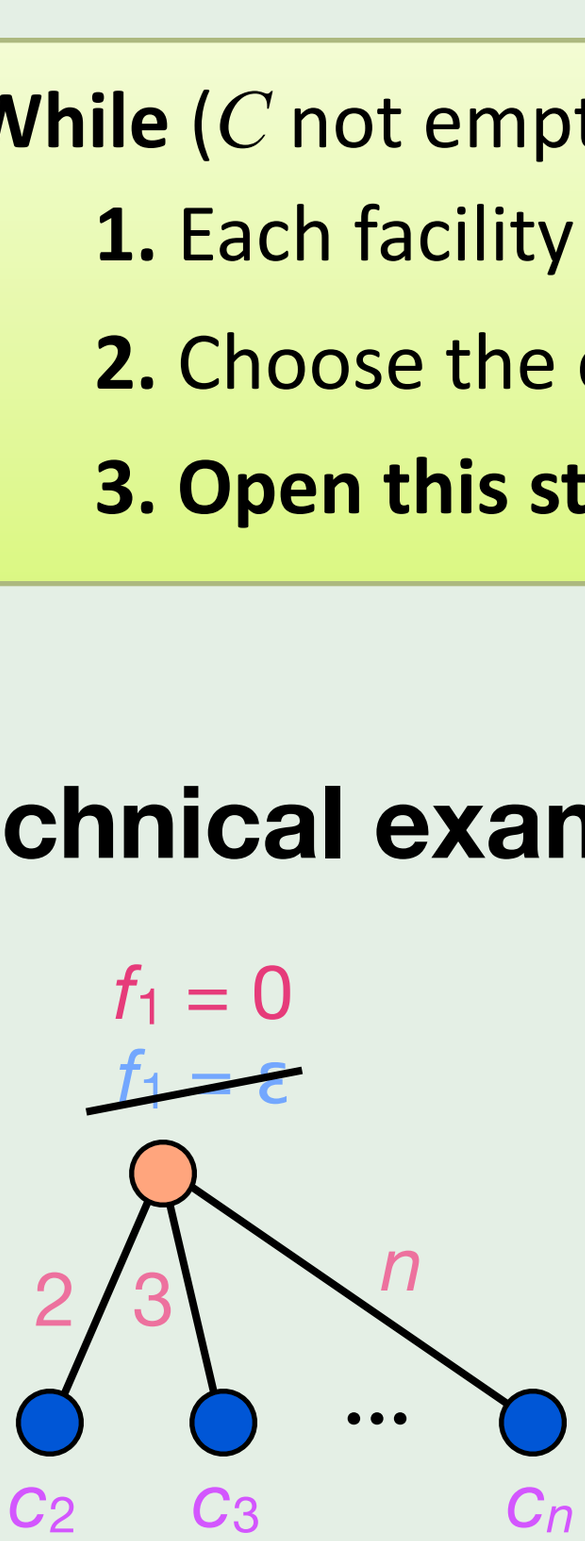
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:

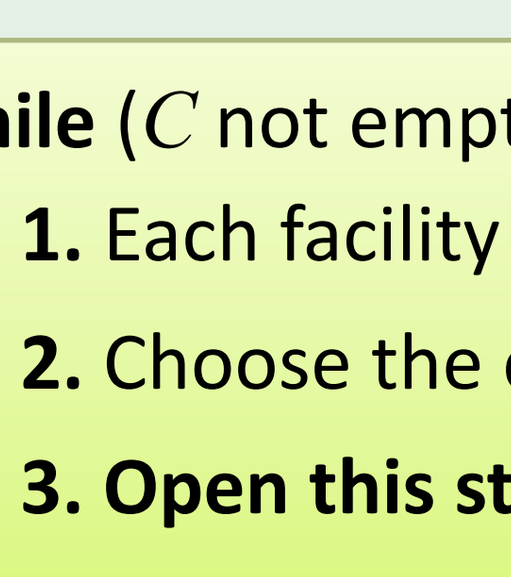
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:

Round 2:

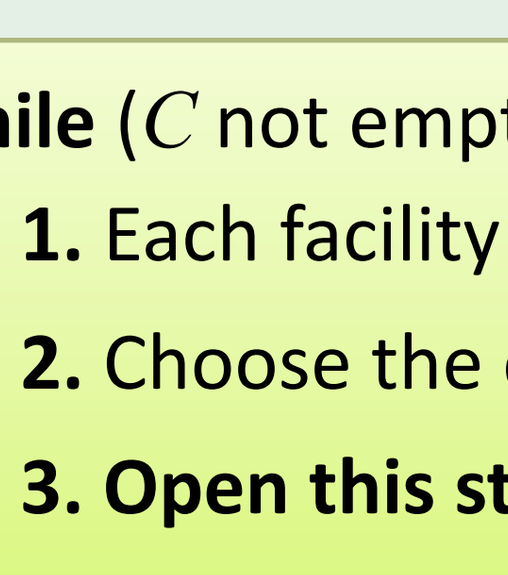
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:

Round 2:

...

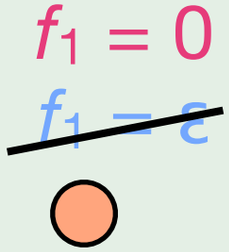
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:

Round 2:

...

Round *n*:

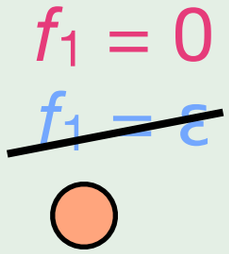
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

a technical example:



n clients, 1 facility

Round 1:  price = 1.5

Round 2:  price = 2

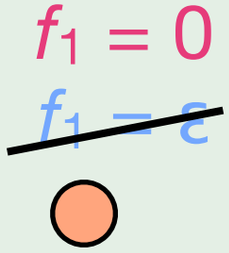
...

Round *n*:  price = n

Observations:

1. **Greedy process** can be ***inherently*** **sequential**
2. **Clients between stars don't overlap**

a technical example:



n clients, 1 facility

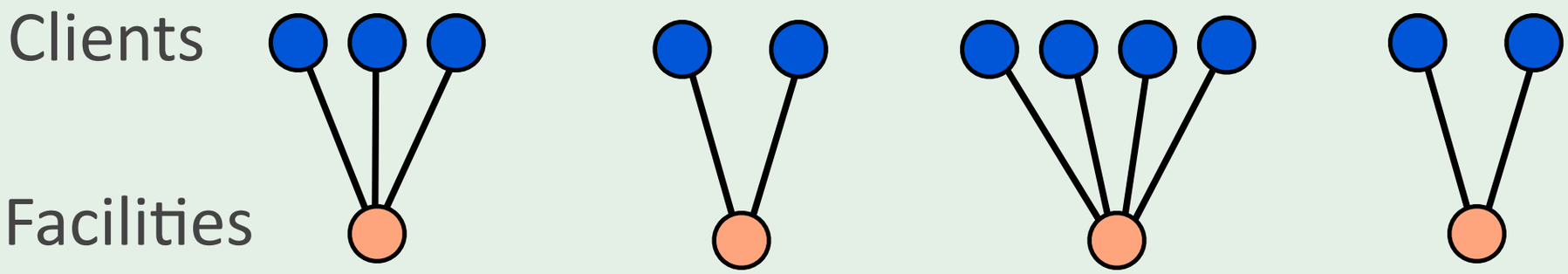
Round 1:  price = 1.5

Round 2:  price = 2

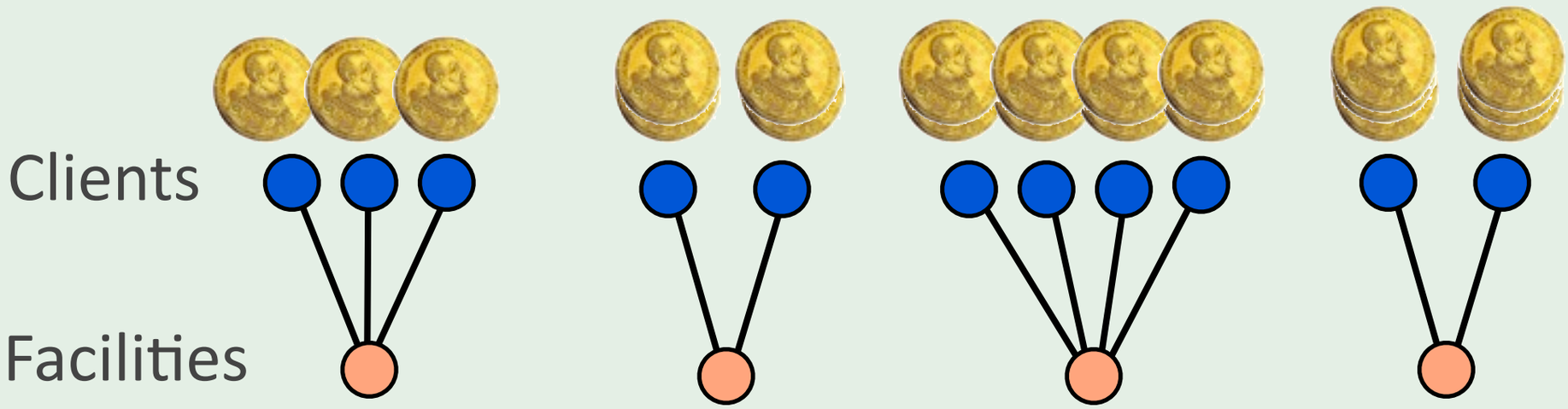
...

Round *n*:  price = *n*

Proof in a Nutshell

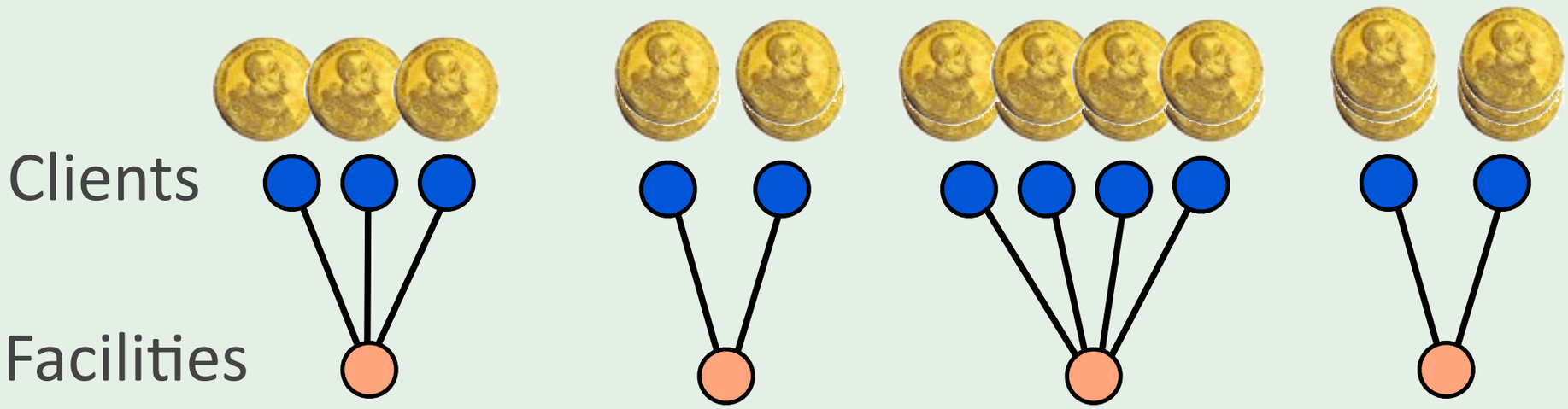


Proof in a Nutshell



For each star *S*, put price(*S*) tokens on each client

Proof in a Nutshell

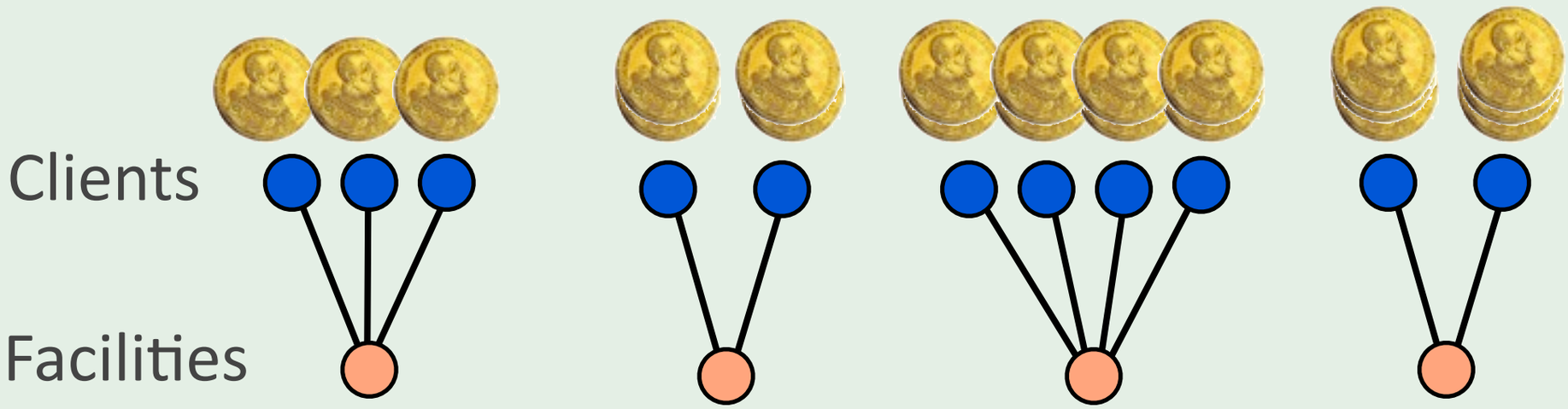


For each star *S*, put price(*S*) tokens on each client

Lemma 1: Facility Cost + Connection Cost ≤ Total #tokens

The stars have no overlapping clients

Proof in a Nutshell



For each star *S*, put price(*S*) tokens on each client

Lemma 1: Facility Cost + Connection Cost ≤ Total #tokens

The stars have no overlapping clients

Lemma 2: Total #tokens ≤ 1.861**OPT**

factor-revealing LP + dual fitting

[Jain et al. ’03]

How to parallelize something that looks inherently sequential?

[Blelloch-T'10, Blelloch-Peng-T'11]

Idea #1: Geometric Scaling

Create opportunities for parallelism

Greedy Algorithm

While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Choose the cheapest star (i, T)

3. Open this star: open i, set $f_i = 0$ and remove T from C

Idea #1: Geometric Scaling

Create opportunities for parallelism

Greedy Algorithm → **Greedier Algorithm**

idea previously used in set cover, vertex cover, ...

While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Suppose the cheapest star has price p

3. GOOD = { star centered at i *if price* ≤ $p(1+\varepsilon)$ }, open them:

Open i, set $f_i = 0$, and remove attached clients from C

Idea #1: Geometric Scaling

Create opportunities for parallelism

Greedy Algorithm → **Greedier Algorithm**

idea previously used in set cover, vertex cover, ...

While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Suppose the cheapest star has price p

3. GOOD = { star centered at i *if price* ≤ $p(1+\varepsilon)$ }, open them:

Open i, set $f_i = 0$, and remove attached clients from C

Good news: ≈ $\log_{1+\varepsilon} m$ rounds; price goes up by $(1 + \varepsilon)$

Problem: Stars Overlap

Opening all "good" stars is too aggressive

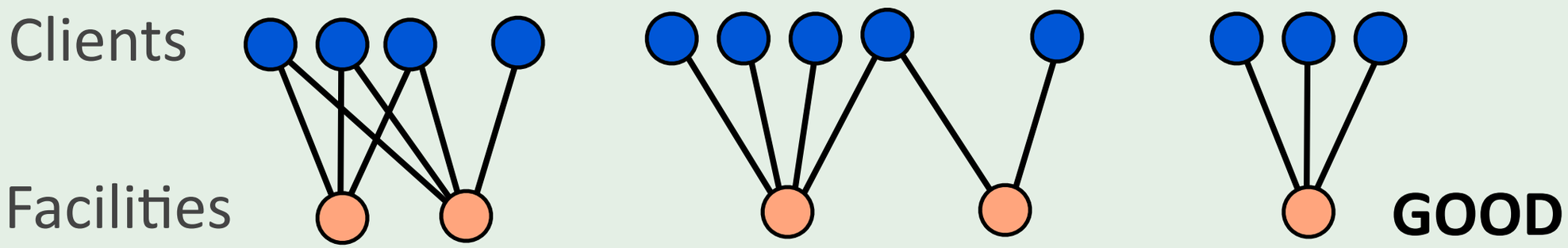
While (C not empty)

1. Each facility i finds the cheapest star centered at i

2. Suppose the cheapest star has cost p

3. GOOD = { star centered at i *if price* ≤ $p(1+\varepsilon)$ }, open them:

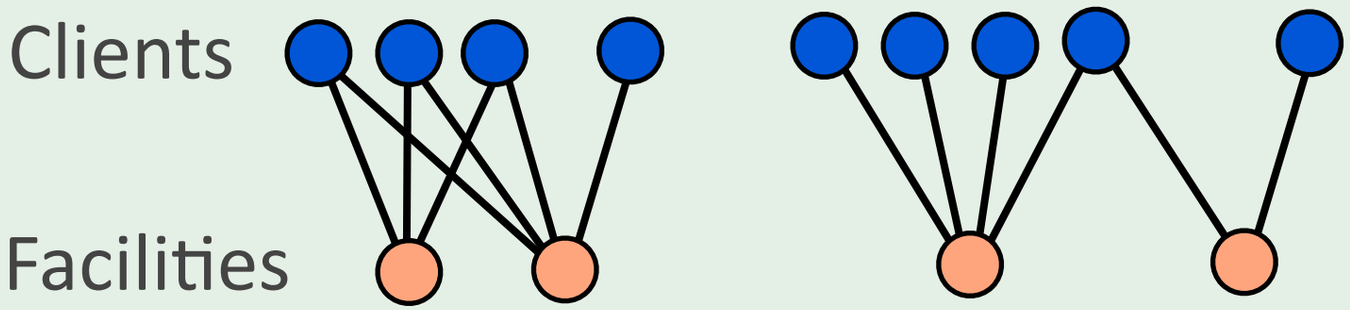
Open i, set $f_i = 0$, and remove attached clients from C



Idea #2: Subselect Facilities

Control how much overlap is allowed

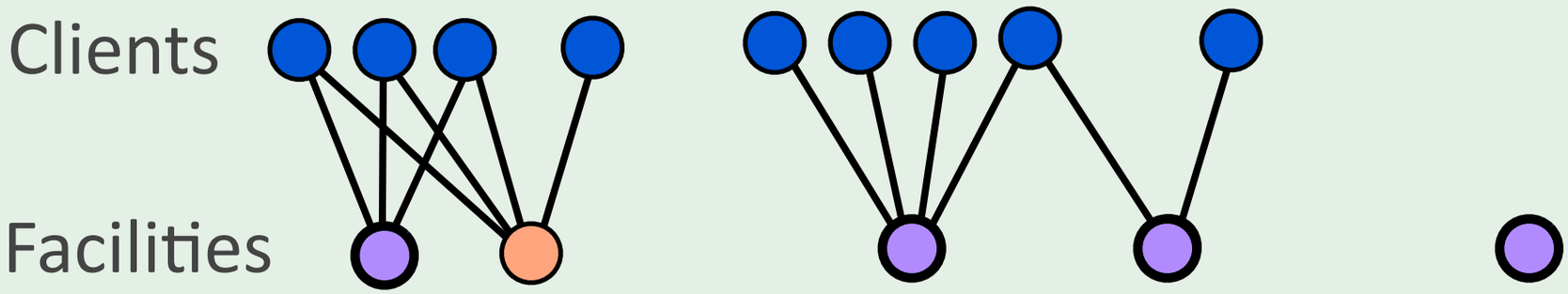
In this round: cheapest star has price ***p***



Idea #2: Subselect Facilities

Control how much overlap is allowed

In this round: cheapest star has price ***p***

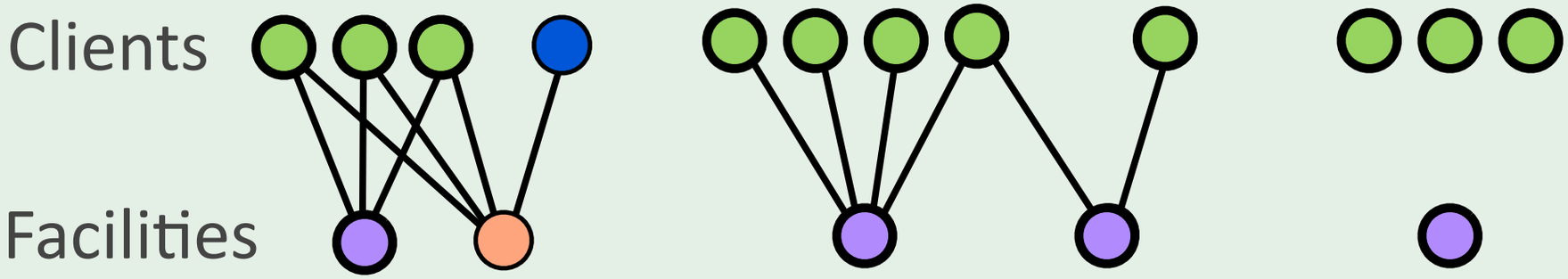


Want: **Select a subset of facilities such that**

Idea #2: Subselect Facilities

Control how much overlap is allowed

In this round: cheapest star has price ***p***

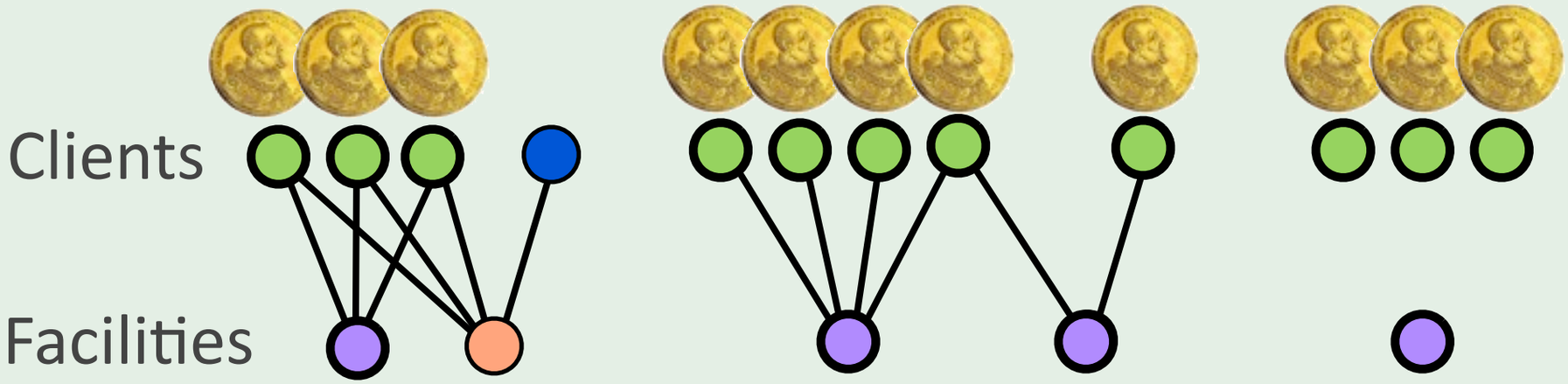


Want: **Select a subset of facilities such that**

Idea #2: Subselect Facilities

Control how much overlap is allowed

In this round: cheapest star has price *p*



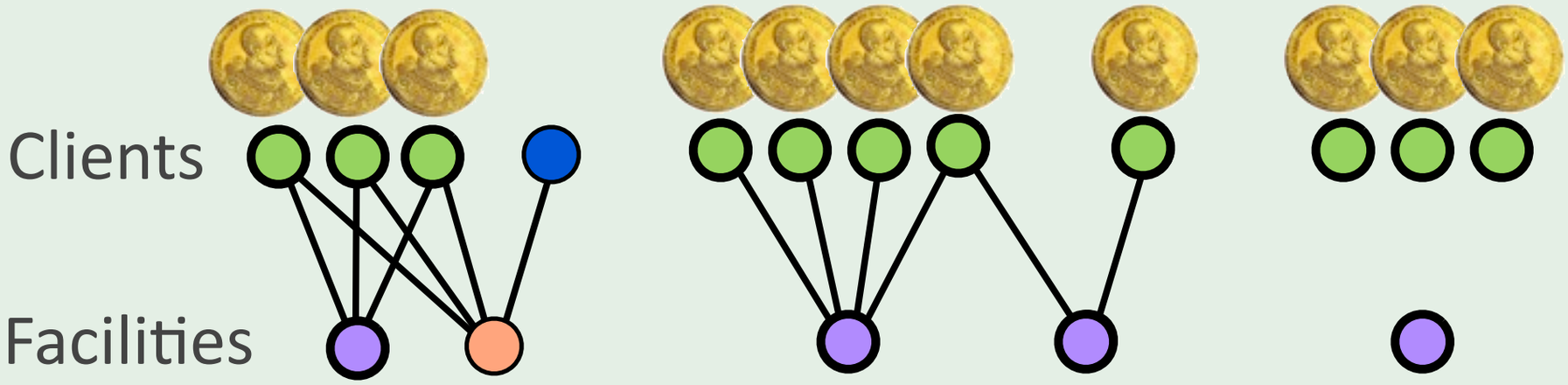
Want: **Select a subset of facilities such that**

if we put *p* tokens on each **"covered" client**

Idea #2: Subselect Facilities

Control how much overlap is allowed

In this round: cheapest star has price *p*



Want: **Select a subset of facilities such that**

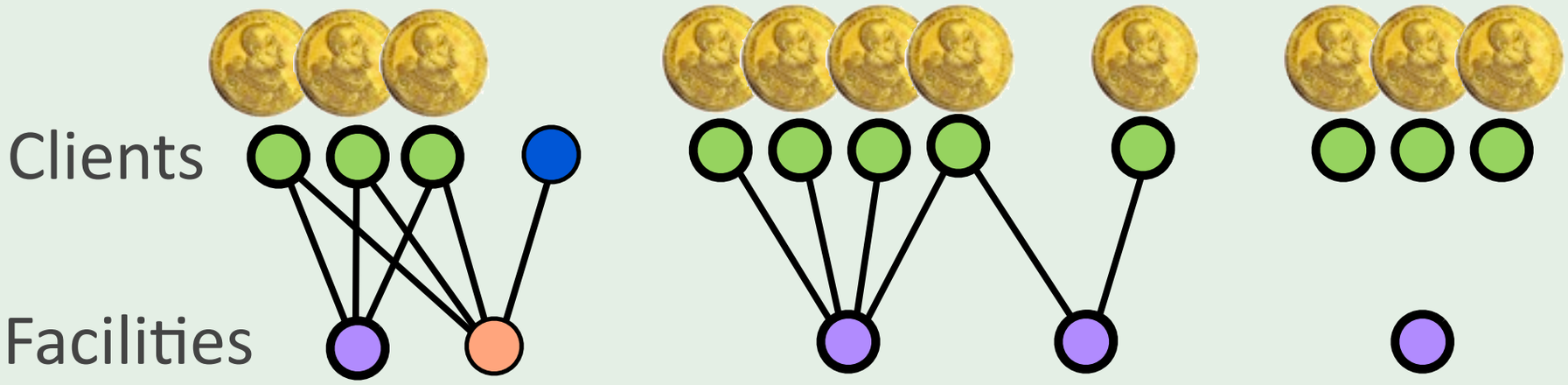
if we put *p* tokens on each **"covered" client**

Property 1: Fac Cost(●) + Conn Cost(●) ≤ (1+δ)Total #tokens

Idea #2: Subselect Facilities

Control how much overlap is allowed

In this round: cheapest star has price *p*



Want: **Select a subset of facilities such that**

if we put *p* tokens on each **"covered" client**

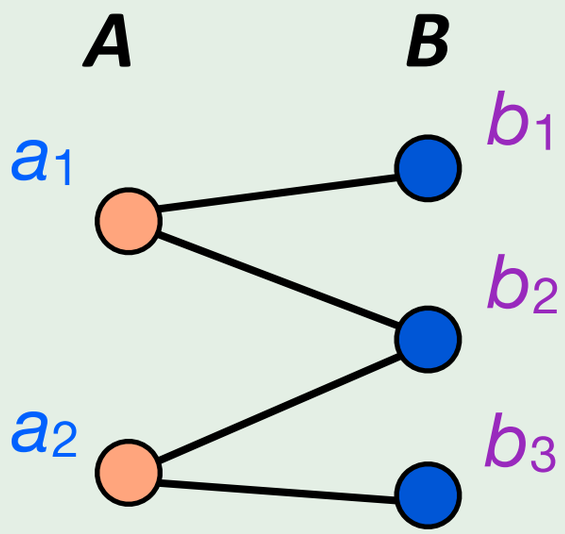
Property 1: Fac Cost(●) + Conn Cost(●) ≤ (1+δ)Total #tokens

Property 2: Price after this round > $p(1+\varepsilon)$

Maximal Nearly Independent Set

Formalizing small overlap and maximality

$N(X)$ = neighbors of X

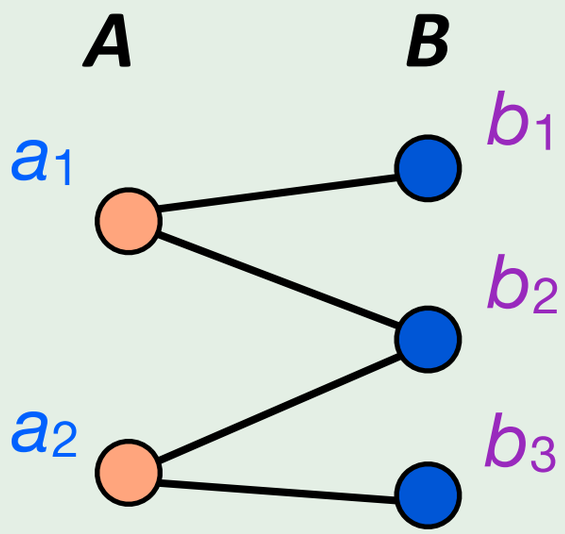


bipartite graph modeling coverage

Maximal Nearly Independent Set

Formalizing small overlap and maximality

$N(X)$ = neighbors of X



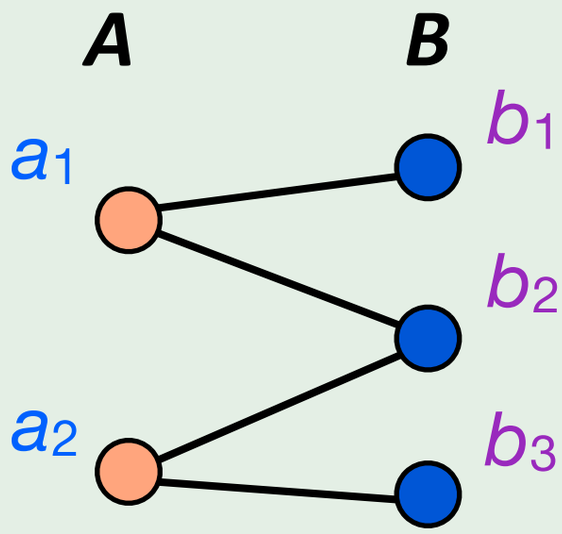
bipartite graph modeling coverage

(ε, δ)**-MaNIS** is $J \subseteq A$ such that

Maximal Nearly Independent Set

Formalizing small overlap and maximality

$N(X)$ = neighbors of X



bipartite graph modeling coverage

(ε, δ)-MaNIS is $J \subseteq A$ such that

1 **small overlaps**

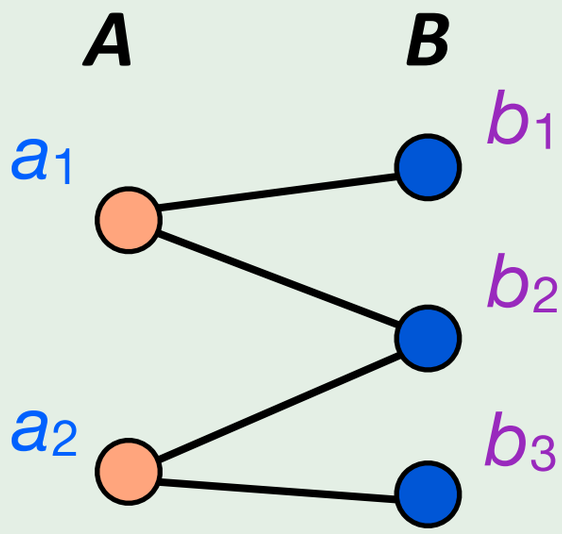
near independence:

$$|N(J)| \geq (1-\delta) \sum_{j \in J} |N(j)|$$

Maximal Nearly Independent Set

Formalizing small overlap and maximality

$N(X)$ = neighbors of X



bipartite graph modeling coverage

(ε, δ)**-MaNIS** is $J \subseteq A$ such that

1 **small overlaps**

near independence:

$$|N(J)| \geq (1-\delta) \sum_{j \in J} |N(j)|$$

2 **"maximal"**

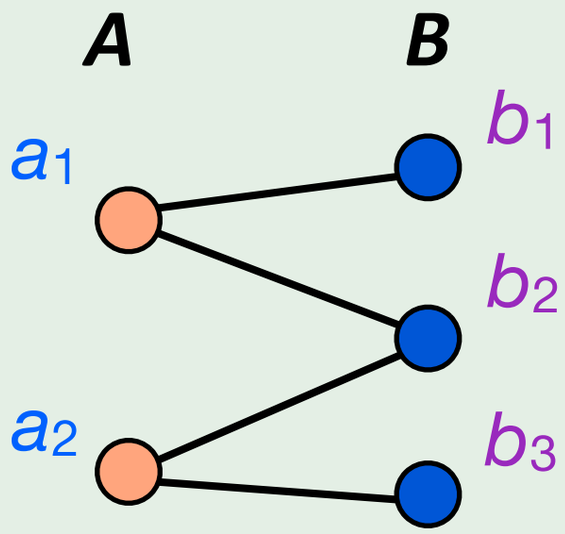
maximality: for all a outside of J,

$$|N(a) \setminus N(J)| < (1-\varepsilon)|N(a)|$$

Maximal Nearly Independent Set

Formalizing small overlap and maximality

$N(X)$ = neighbors of X



bipartite graph modeling coverage

(ε, δ)**-MaNIS** is $J \subseteq A$ such that

1 **small overlaps**

near independence:

$$|N(J)| \geq (1-\delta) \sum_{j \in J} |N(j)|$$

2 **"maximal"**

maximality: for all a outside of J,

$$|N(a) \setminus N(J)| < (1-\varepsilon)|N(a)|$$

1. not unique
2. $\varepsilon = \delta = 0$ no overlap —> maximal set packing
3. simple $O(|E|)$ seq. alg

Back to Facility Location

Lemma: If we can compute (ε, δ)-MaNIS, then we have a $1.861/(1-\varepsilon-\delta)$-approx.

Back to Facility Location

Lemma: If we can compute (ε, δ)-MaNIS, then we have a $1.861/(1-\varepsilon-\delta)$-approx.

Lemma 1*:

Facility Cost + Connection Cost ≤ Total #tokens/$(1-\delta)$

The stars have almost no overlapping clients

Back to Facility Location

Lemma: If we can compute (ε, δ)-MaNIS, then we have a $1.861/(1-\varepsilon-\delta)$-approx.

Lemma 1*:

Facility Cost + Connection Cost ≤ Total #tokens/$(1-\delta)$

The stars have almost no overlapping clients

Lemma 2*: Total #tokens ≤ 1.861**OPT/**$(1-\varepsilon)$

factor-revealing LP + dual fitting

[Jain et al. '03]

+ geometric scaling

How to Compute MaNIS?

Implicit in algorithms from previous work

$$m = |F| \times |C|$$

How to Compute MaNIS?

Implicit in algorithms from previous work

$m = |F| \times |C|$

Berger, Rompel, and Shor'94
(also Chierichetti, Kumar, and Tomkins'10) → (ε, 8ε)-MaNIS

RNC $O(m \log^4 m)$-work (1.861 + ε)-approx

How to Compute MaNIS?

Implicit in algorithms from previous work

$m = |F| \times |C|$

Berger, Rompel, and Shor'94
(also Chierichetti, Kumar, and Tomkins'10) → (ε, 8ε)-MaNIS

RNC $O(m \log^4 m)$-work (1.861 + ε)-approx

Rajagopalan and Vazirani'98 → (ε, ½+ ε)-MaNIS

RNC $O(m \log^2 m)$-work (3.722 + ε)-approx

How to Compute MaNIS?

Implicit in algorithms from previous work

$m = |F| \times |C|$

Berger, Rompel, and Shor'94
(also Chierichetti, Kumar, and Tomkins'10) → (ε, 8ε)-MaNIS

RNC $O(m \log^4 m)$-work $(1.861 + \varepsilon)$-approx

Rajagopalan and Vazirani'98 → (ε, ½+ ε)-MaNIS

RNC $O(m \log^2 m)$-work $(3.722 + \varepsilon)$-approx

Next step: Linear work for any value of ε?

How to Compute MaNIS?

Implicit in algorithms from previous work

$m = |F| \times |C|$

Berger, Rompel, and Shor'94
(also Chierichetti, Kumar, and Tomkins'10) → (ε, 8ε)-MaNIS

RNC $O(m \log^4 m)$-work $(1.861 + \varepsilon)$-approx

Rajagopalan and Vazirani'98 → (ε, ½+ ε)-MaNIS

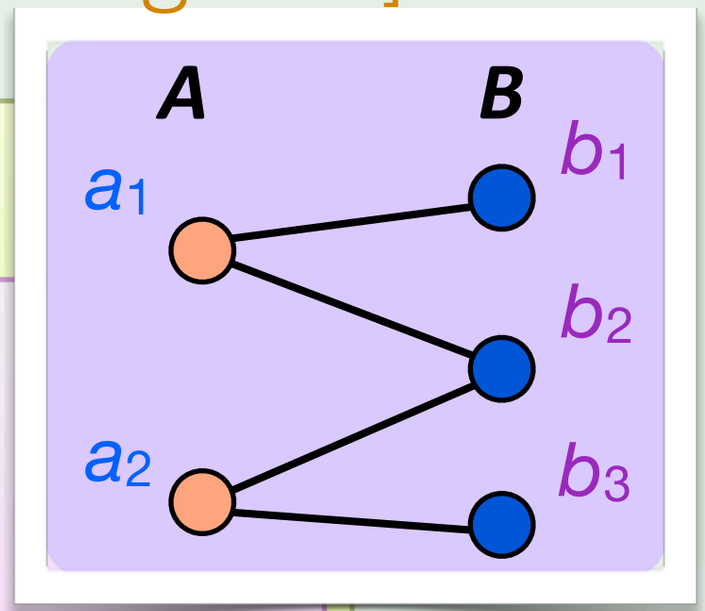
RNC $O(m \log^2 m)$-work $(3.722 + \varepsilon)$-approx

Next step: Linear work for any value of ε?

RNC $O(m \log m)$-work $(1.861 + \varepsilon)$-approx

A Simple Linear-Work MaNIS

$O(\log^2 |E|)$-depth MaNIS [Blelloch-Peng-T'11]



While (*A* not empty)

a) Pick a random permutation π of ***A***

b) Each $b \in$ ***B*** joins the highest π-ranked nbr

c) For each $a \in$ ***A***, if (1 - δ) fraction of nbrs joined it, add a to output and remove a's nbr

d) Remove $a \in$ ***A*** if degree less than (1-ε) fraction of its original degree

Idea: random permutation removes a const fraction of edges takes O(log $|E|$) rounds

Parallel Greedy Facility Location

Putting things together

Parallel Greedy Facility Location

Putting things together

- **Idea #1: Geometric Scaling**

  outer loop: mimic greedy behavior

  price goes up by (1 + ε), so O(log m) rounds

Parallel Greedy Facility Location

Putting things together

- **Idea #1: Geometric Scaling**

  outer loop: mimic greedy behavior

  price goes up by (1 + ε), so O(log m) rounds

- **Idea #2: Subselection**

  polylog depth and *O*(*m*) work, ***whp.***

Parallel Greedy Facility Location

Putting things together

- **Idea #1: Geometric Scaling**

  outer loop: mimic greedy behavior

  price goes up by (1 + ε), so O(log m) rounds

- **Idea #2: Subselection**

  polylog depth and *O*(*m*) work, ***whp.***

Plus, additional O(log m) depth, O(m) work basic operations in the outer loop.

Parallel Greedy Facility Location

Putting things together

- **Idea #1: Geometric Scaling**

  outer loop: mimic greedy behavior

  price goes up by (1 + ε), so O(log m) rounds

- **Idea #2: Subselection**

  polylog depth and *O*(*m*) work, ***whp.***

Theorem:

RNC O(*m* log *m*)-work, factor-(1.861+ε) *greedy-style* approximation algorithm.

Parallel Greedy Facility Location

Putting things together

- **Idea #1: Geometric Scaling**

  outer loop: mimic greedy behavior

  price goes up by (1 + ε), so O(log m) rounds

- **Idea #2: Subselection**

  polylog depth and *O*(*m*) work, ***whp.***

Theorem:

RNC O(*m* log *m*)-work, factor-(1.861+ε) *greedy-style* approximation algorithm.

Using MaNIS: Linear-work algorithms for max cover, (weighted) set cover, min-sum set cover

Take-Home Points

Maximal Nearly Independent Set

Pick a maximal collection that has small overlap

... more at SPAA'11

Linear-Work Greedy Parallel Approximation Algorithms for Set Covering and Variants

Acknowledgments:

Guy Blelloch, Anupam Gupta, Ioannis Koutis, Gary Miller, Richard Peng

Take-Home Points

Thank you!

Maximal Nearly Independent Set

Pick a maximal collection that has small overlap

... more at SPAA'11

Linear-Work Greedy Parallel Approximation Algorithms for Set Covering and Variants

Acknowledgments:

Guy Blelloch, Anupam Gupta, Ioannis Koutis, Gary Miller, Richard Peng

Take-Home Points

Thank you!

Maximal Nearly Independent Set

Pick a maximal collection that has small overlap

... more at SPAA'11

Linear-Work Greedy Parallel Approximation Algorithms for Set Covering and Variants

Shameless Plug

Near-Linear Work SDD Solver

Solve $Ax = b$ in $\tilde{O}(\#nnz)$-work $O(\#nnz^{1/3})$-depth

if A is symmetric diagonally dominant (SDD)

... more at SPAA'11

Near Linear-Work Parallel SDD Solvers, Low-Diameter Decomposition, and Low-Stretch Subgraphs

Acknowledgments:

Guy Blelloch, Anupam Gupta, Ioannis Koutis, Gary Miller, Richard Peng