

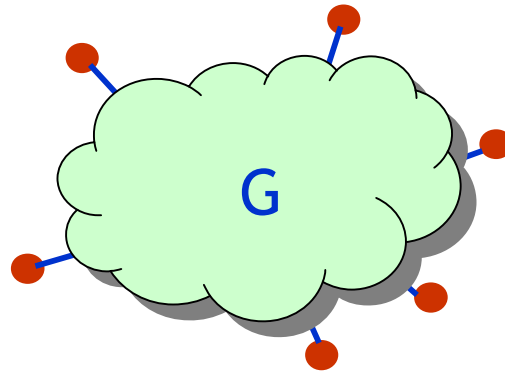
building networks without the traffic matrix

Anupam Gupta

Carnegie Mellon University

(work done at Cornell & Bell Labs)

how to buy bandwidth in a network



Base network G

1. Estimate traffic volume $D(u,v)$ between all u and v
2. Exact or approximate algorithm to find cheap sub-network

Step #2 -- tons of great algorithms and counting...

Step #1?

possible approaches

- Can estimate all $D(u,v)$

Upper bound on traffic between terminals u and v

- $O(k^2)$ values
 - Must look at traffic content
 - many problems
 - conservative - may not handle volatility
- Much cheaper to measure “marginals”

Upper bound on amount of traffic terminal v is involved in

Call it $b(v)$

universal guarantees

Want to build cheapest network
that allows routing of any “valid” demand D

$$D \text{ valid} \Leftrightarrow \sum_v D(u, v) \leq b(u)$$

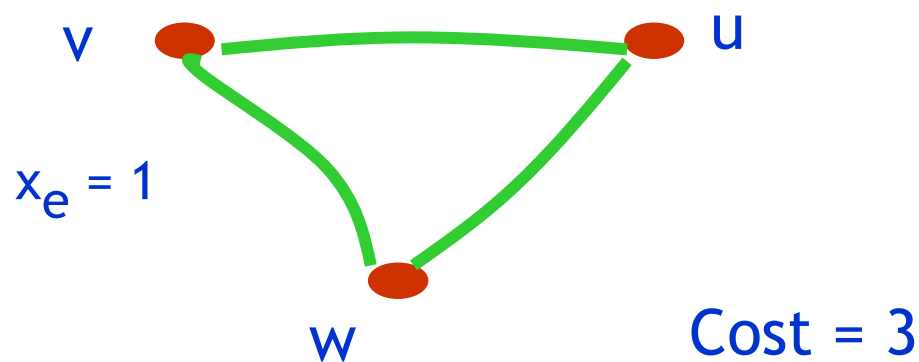
Why this?

- worst-case guarantees
- handles volatility of traffic patterns
- good, if no additional information

an example

- Suppose we measured $D(u,v) = D(u,w) = D(v,w) = 1$

- Conventional solution:

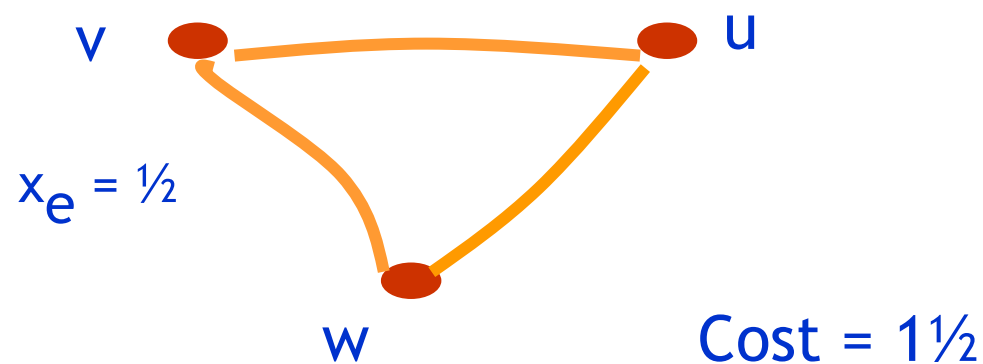


an example

- Suppose we measured $D(u,v) = D(u,w) = D(v,w) = 1$

but $b(x) = 1$ for each vertex x

- So this suffices:



Cost of solutions

- Work in linear edge cost model
 - Edge e has cost c_e per unit of bandwidth
 - allocating x_e bandwidth on e costs $c_e x_e$
 - No capacities on edges

the model

- Given:
 - Set A of k terminals
 - Terminal thresholds $b(v)$
 - Edge costs c_e
- Output:
 - Build cheapest network that supports any valid demand D

$$D \text{ valid} \Leftrightarrow \sum_v D(u, v) \leq b(u)$$

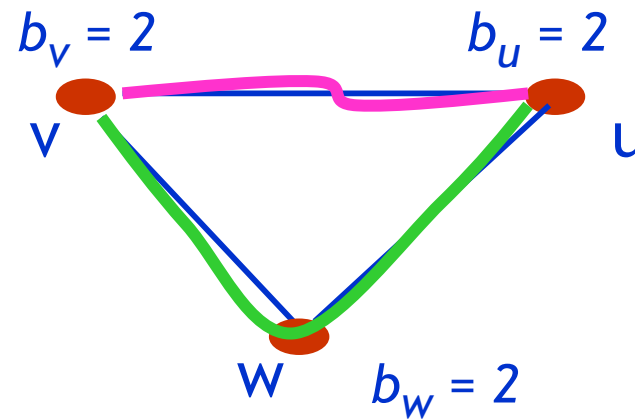
History

- Considered for broadband ATM networks in early '90s
[Fingerhut Suri Turner, JAlgos'97]
- VPN "hose" model by
[Duffield Goyal Greenberg Mishra
Ramakrishnan vanderMerwe, SIGCOMM'99]
- This talk based on some results from
[G. Kleinberg Kumar Rastogi Yener, STOC'01]
[G. Kumar Roughgarden, STOC '03]

other properties of the network

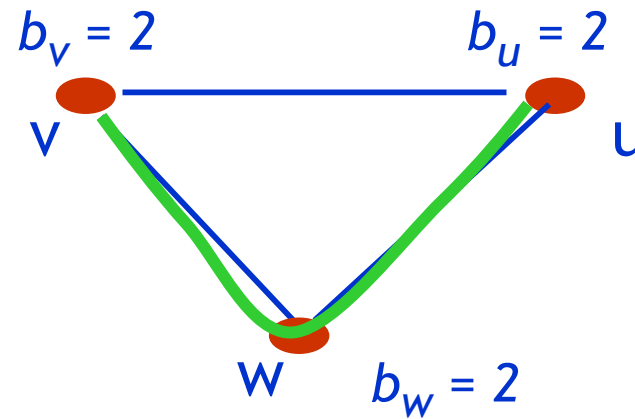
- Scalability
 - Want sparse network
- Routability
 - Want good routing protocols
- Non-blocking algorithms

Non-blocking?



- $x_e = 1$ for each edge e
- Suppose we start with $D(v,u) = 2$, all else 0.
Now $D(v,u)$ falls to 1.

Non-blocking?



- $x_e = 1$ for each edge e
- Suppose we start with $D(v,u) = 2$, all else 0.
Now $D(v,u)$ falls to 1. How do we route a new $D(w,u) = 1$?

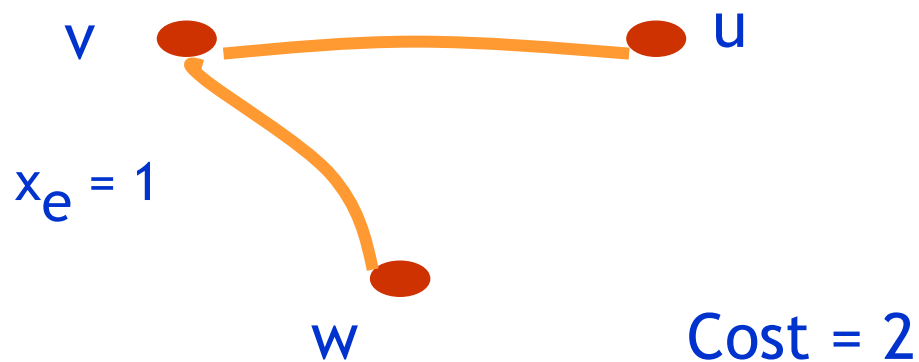
other properties of the network

- Scalability
 - Want sparse network
- Routability
 - Want good routing protocols
- Non-blocking algorithms
- Unsplittable routing
 - Want all **u-v** traffic to take same path

All these properties achieved by trees.

an example

- Suppose we measured $D(u,v) = D(u,w) = D(v,w) = 1$
but $b(x) = 1$ for each vertex x
- For tree solution, this suffices:



the objective

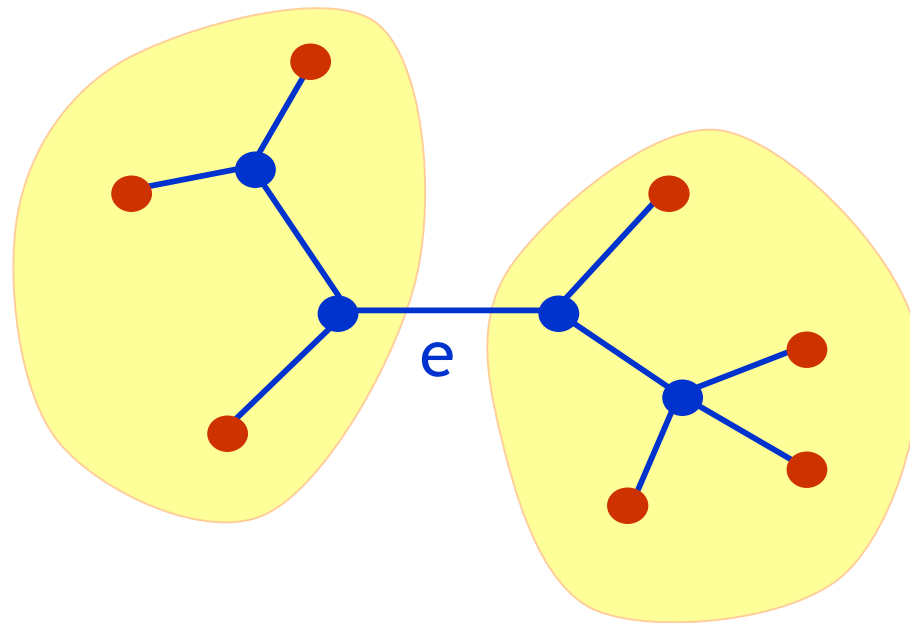
- Build the cheapest network
 - that supports any valid demand D
 - allocated edges form a tree.

- Fact #1

We can find the optimal tree

- Assume: $b(v) = 1$ for all v .

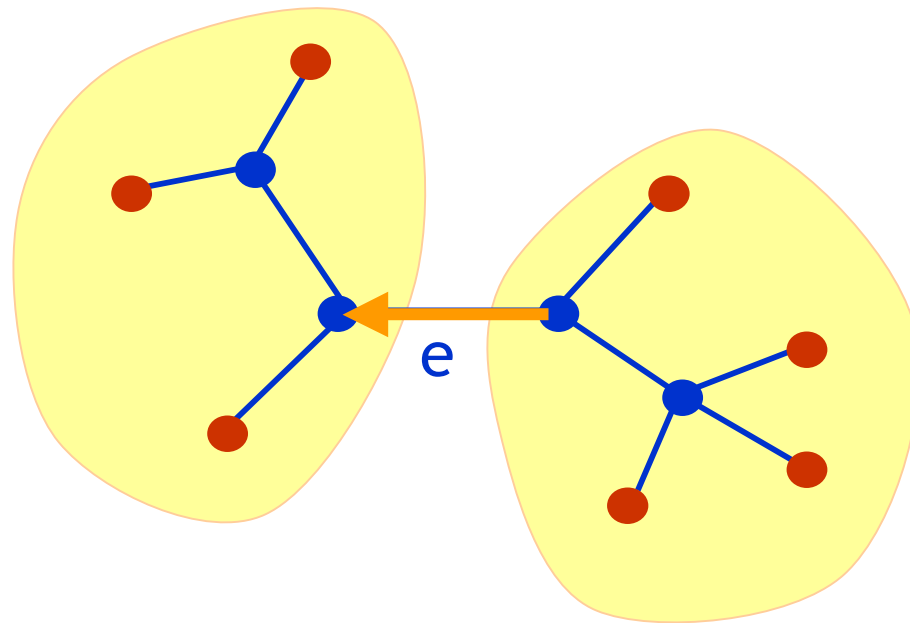
Proof



Look at e in optimal tree T

$$x(e) = \min \{ \#terminals(Left), \#terminals(Right) \}$$

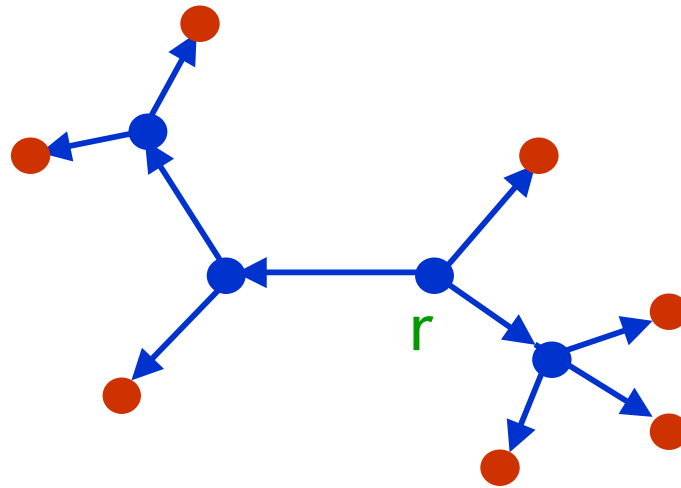
Proof



Look at e in optimal tree T

$$x(e) = \min \{ \underline{\#terminals(Left)}, \#terminals(Right) \}$$

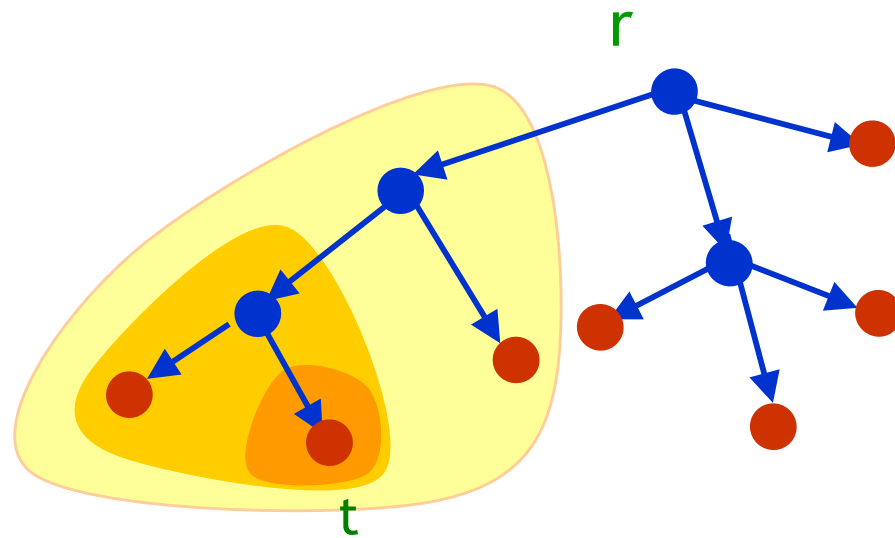
Proof



Gives us a directed tree

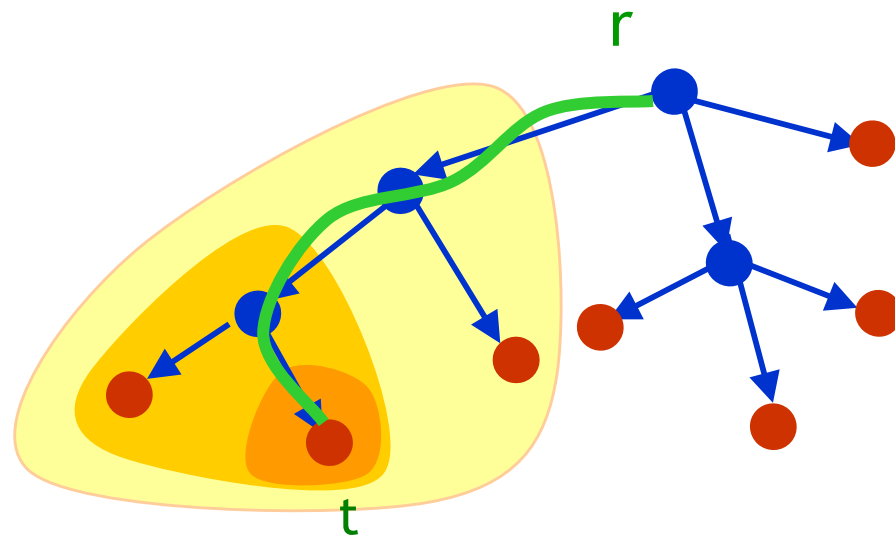
Fact: this tree has unique source r

Proof



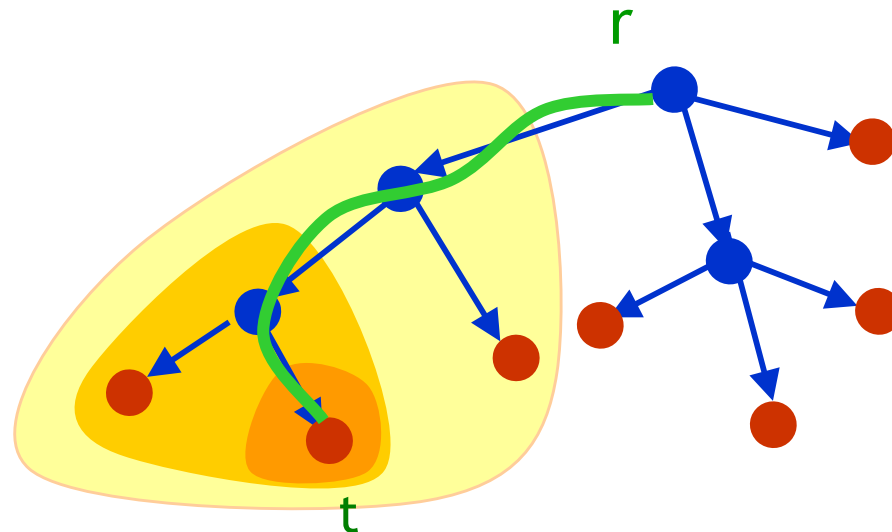
$$\text{Cost}(T) = \sum_e c_e \#(\text{terminals in subtree below } e)$$

Proof



$$\begin{aligned}\text{Cost}(T) &= \sum_e c_e \#(\text{terminals in subtree below } e) \\ &= \sum_{t \in A} \sum_{e \in \text{Path}(r, t)} c_e\end{aligned}$$

Proof



$$\begin{aligned}\text{Cost}(T) &= \sum_e c_e \#(\text{terminals in subtree below } e) \\ &= \sum_{t \in A} \sum_{e \in \text{Path}(r,t)} c_e \\ &\geq \sum_{t \in A} \text{shortest } (r,t) \text{ path} \\ &\quad (\text{Imagining the } c_e \text{'s to be distances})\end{aligned}$$

Results

- Fact #1

We can find the optimal tree

- all-pairs shortest path computation

Comparison with a proportional demand matrix

- Imagine each terminal pair u,v talking at rate
 $D(u,v) = 1/(k-1)$
- Let best network for this fixed matrix have cost = Symm^*
- Theorem: [FST '97, GKKRY '01]

$$\text{Symm}^* \leq \text{OPT}(\text{tree}) \leq 2 \text{Symm}^*$$

Results

- Fact #1

We can find the optimal tree

- all-pairs shortest paths computation
- At most twice cost of optimal solution
for proportional demands

How much cheaper can non-trees be?

- build cheapest network such that

\forall valid D

\exists a multicommodity flow f

f_{uv} satisfies $D(u,v)$

- This network must support: $D(u,v) = 1/(k-1)$
- $\text{Symm}^* \leq \text{this cost} \leq \text{OPT} \leq 2 \text{Symm}^*$

Results

- Fact #1

We can find the optimal tree

- all-pairs shortest paths computation
- At most twice cost of optimal solution
for proportional demands
- Dropping unsplittability/non-blocking can only halve the cost

open problem

- Build cheapest network such that

$\forall D \in \mathcal{P}$ (e.g., $D \in$ valid demands)
 \exists a multicommodity flow f
 f_{uv} satisfies $D(u,v)$

Problem: Find this optimally?

- LP approaches don't seem to work
 - Co-NP hard for directed graphs
 - Poly-time if \mathcal{P} has few extreme points

A (slightly) richer model

A tale of two thresholds

- Each terminal has two thresholds:

- Ingress threshold : $b\text{-in}(v)$
 - Egress threshold : $b\text{-out}(v)$
- Let $b(v) = b\text{-in}(v) + b\text{-out}(v)$

- Demand matrix D is *valid* iff

$$\sum_v D(u, v) \leq b\text{-out}(u) \quad \&\& \quad \sum_u D(u, v) \leq b\text{-in}(v)$$

- Assume: Each terminal is

- sender [$b\text{-in} = 0, b\text{-out} = 1$]
- receiver [$b\text{-in} = 1, b\text{-out} = 0$]

Results

- Theorem #2

Finding optimal tree is max-SNP hard

- Theorem #3

We can find a tree with cost within $4 \times \text{OPT}(\text{tree})$

- Theorem #4

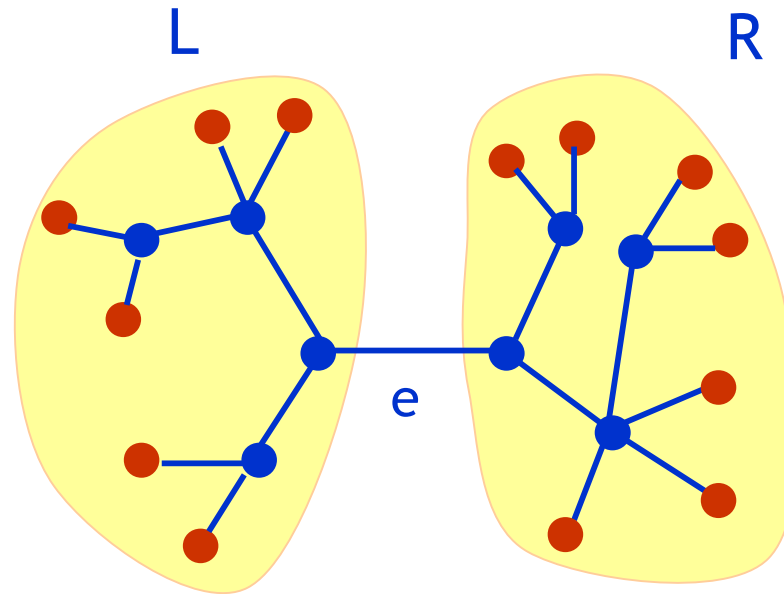
We can find a tree with cost within $6 \times \text{OPT}(\text{graph})$

Results

- Theorem #3

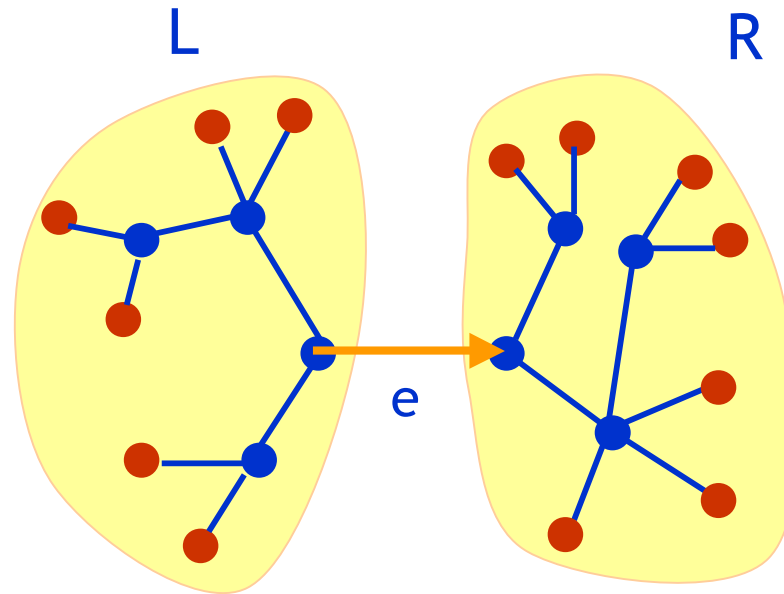
We can find a tree with cost within $4 \times \text{OPT}(\text{tree})$

Structure of solutions



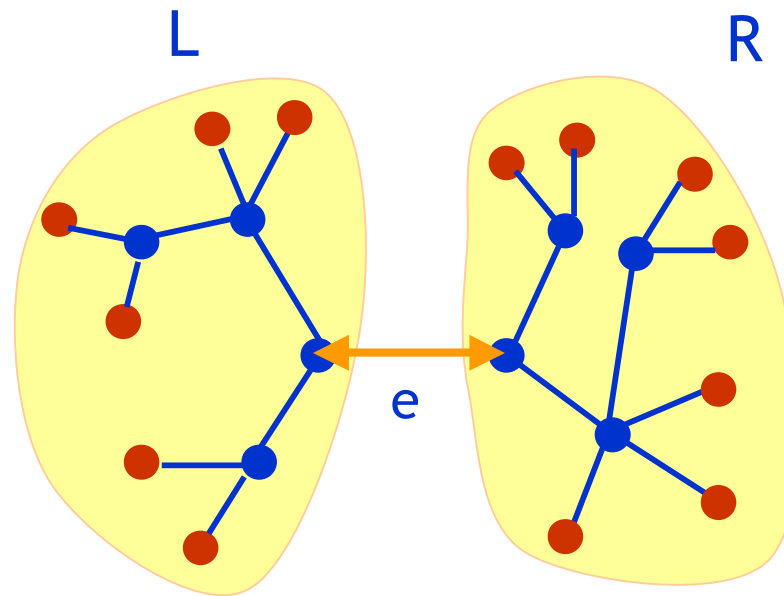
$$x(e) = \min \{ \#senders(L), \#receivers(R) \} \\ + \min \{ \#receivers(L), \#senders(R) \}$$

Structure of solutions



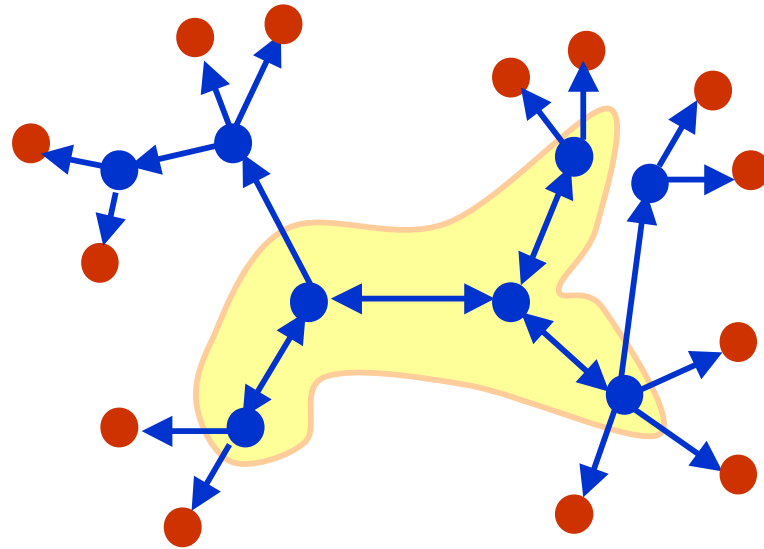
$$\begin{aligned} x(e) &= \min \{ \#senders(L), \#receivers(R) \} \\ &\quad + \min \{ \#receivers(L), \#senders(R) \} \\ &= \#terminals(R) \end{aligned}$$

Structure of solutions



$$\begin{aligned} x(e) &= \min \{ \# \underline{\text{senders}}(L), \# \text{receivers}(R) \} \\ &\quad + \min \{ \# \text{receivers}(L), \# \underline{\text{senders}}(R) \} \\ &= \# \text{senders}(V) \end{aligned}$$

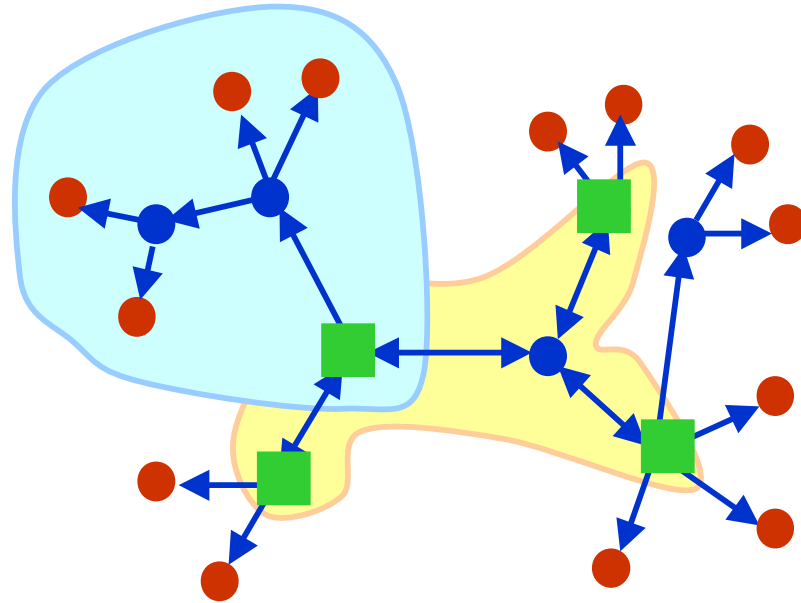
Structure of solutions



Core connected by double arrows

Recall: $x(e) = \#\text{senders}(V) \quad \forall e \text{ in core}$

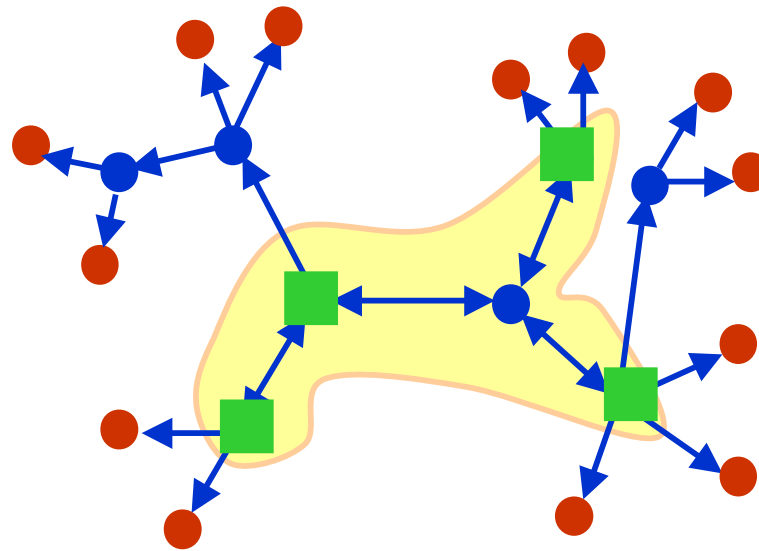
Structure of solutions



Edge nodes: branchings hang off

Branchings are shortest-path trees from edge-node to terminals

but note that...

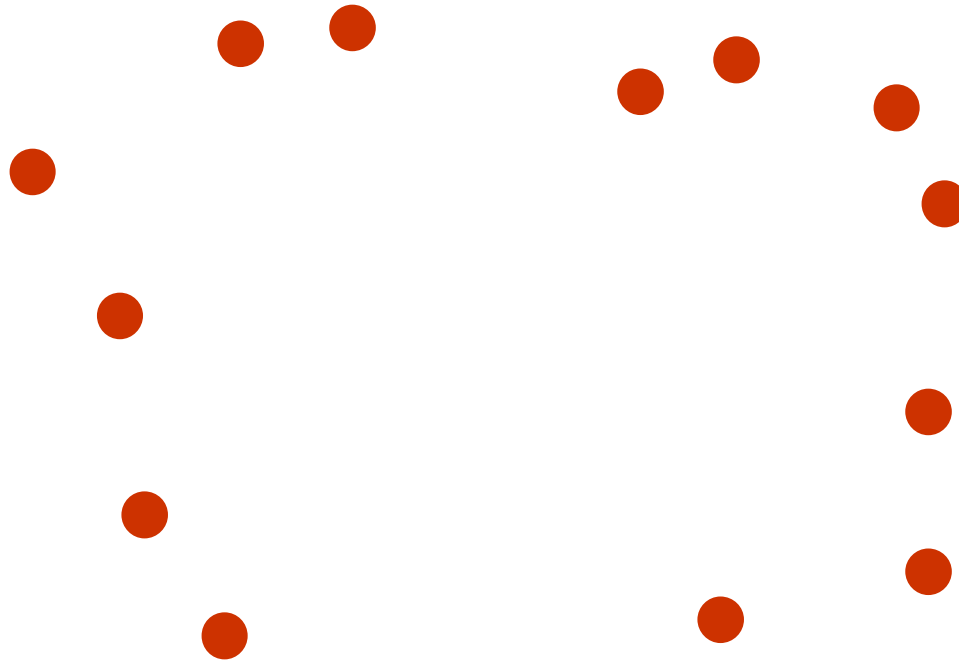


Note: edges nodes completely specify the solution

and now for something completely different...

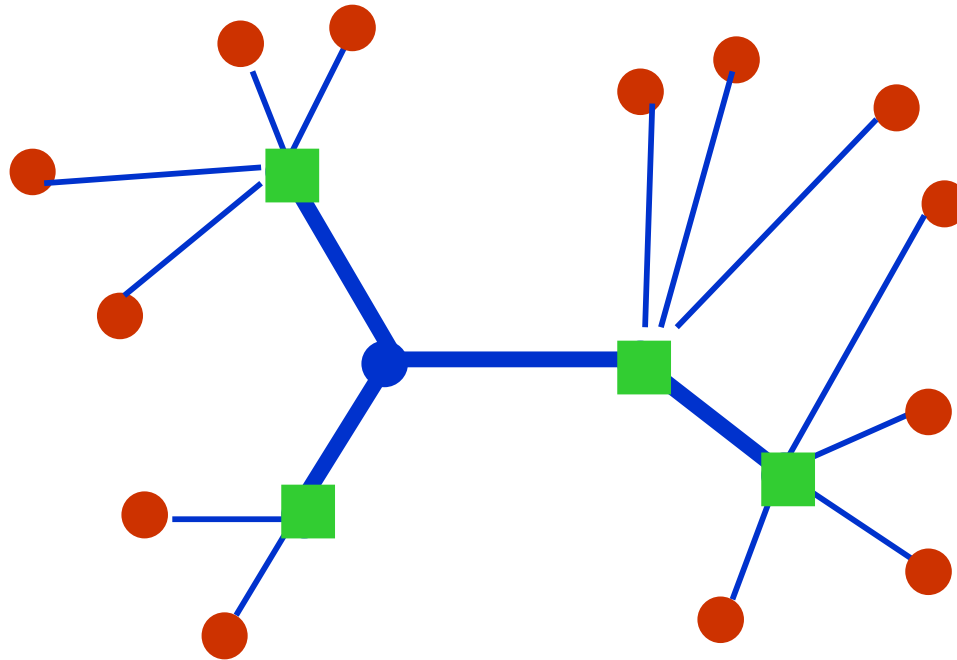
(well, not quite)

connected facility location



given demand set D in a metric space
(assume j has demand d_j for all $j \in D$)

connected facility location



locate facilities F to minimize

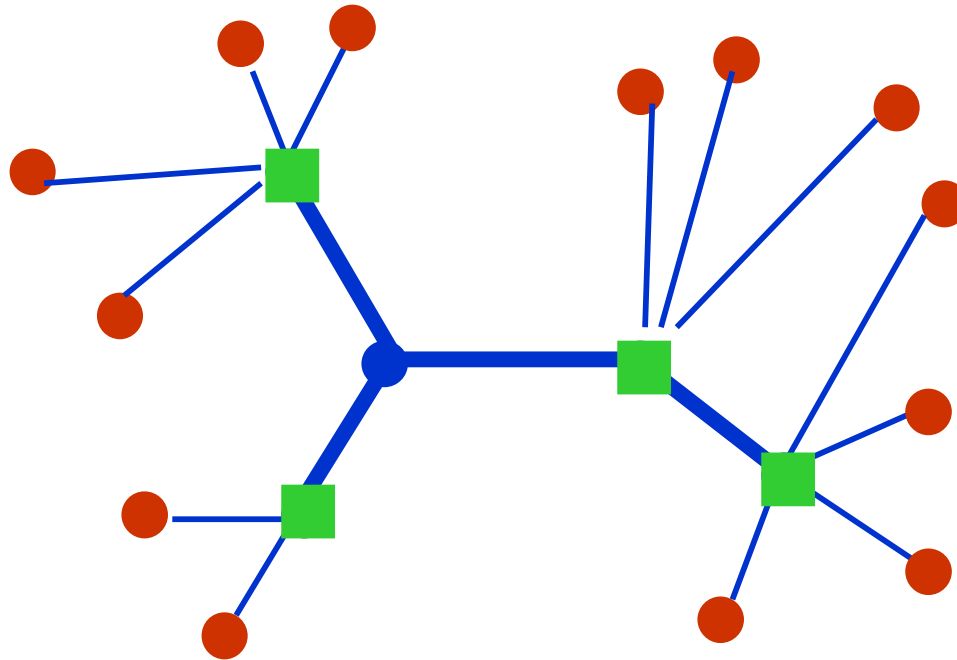
$$\sum_j d_j \times \text{distance}(i^*_j, j)$$

$b(j)$

$$+ M \times \text{Steiner}(F)$$

$\min\{\#\text{senders}(V), \#\text{receivers}(V)\}$

a.k.a. "Rent-or-Buy"



locate facilities F to minimize

$$\sum_j d_j \times \text{distance}(i^*_j, j)$$

$b(j)$

$$+ M \times \text{Steiner}(F)$$

$\min\{\#\text{senders}(V), \#\text{receivers}(V)\}$

Results

- Theorem #3

We can find a tree with cost within $4 \times \text{OPT}(\text{tree})$

Proof:

Finding $\text{OPT}(\text{tree})$ is equivalent to
Connected Facility Location (ConnFL)

now, to give an algorithm for ConnFL

algorithms for connFL

- via LP rounding
 - [Ravi Salman ESA '99]
 - [G. Kleinberg Kumar Rastogi Yener STOC '01]
- via reductions to classical facility location
 - [Karger Minkoff FOCS '00]
 - [Guha Meyerson Munagala FOCS '00]
- primal-dual schema
 - [Swamy Kumar APPROX '02]

algorithm SimpleConnFL

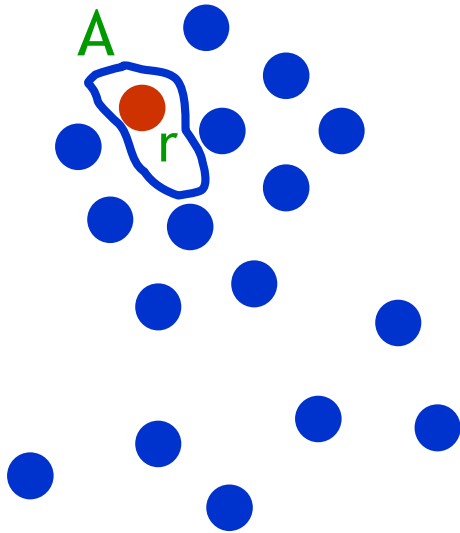
- 1) Mark each demand with probability $1/M$
(demands in this set F are our facilities)
- 2) Build MST on facilities F
- 3) Connect demands to closest facility

Theorem: SimpleCFL is a 4-approximation algorithm

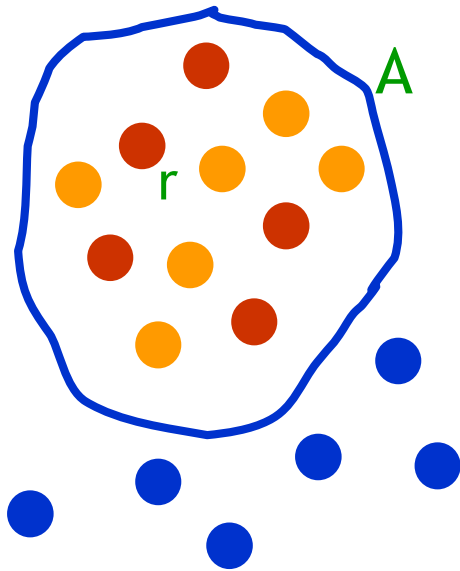
[G. Kumar Roughgarden STOC '03]

an incremental view

- Init: Pick a "root" r , $A = \{r\}$



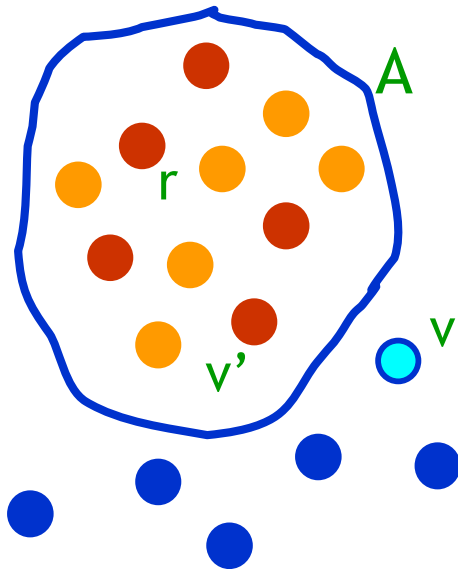
an incremental view



- Init: Pick a “root” r , $A = \{r\}$
- At step t
Pick vertex v in $D - A$ closest to a red vertex v'

flip a coin ($p = 1/M$) for it

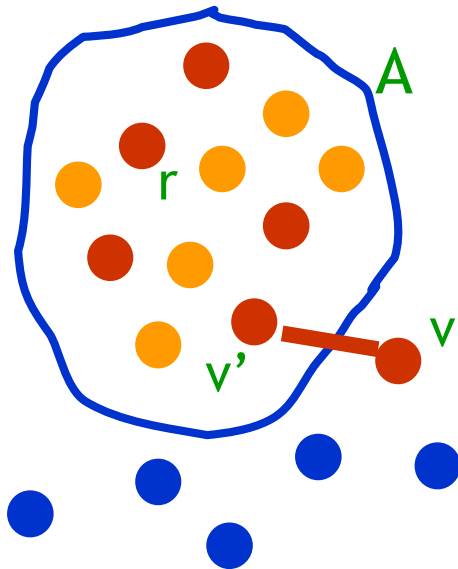
an incremental view



- Init: Pick a “root” r , $A = \{r\}$
- At step t
Pick vertex v in $D - A$ closest to a red vertex v'

flip a coin ($p = 1/M$) for it

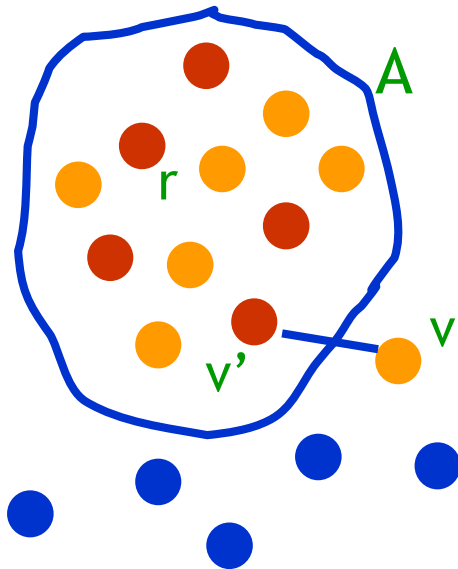
an incremental view



- Init: Pick a “root” r , $A = \{r\}$
- At step t
Pick vertex v in $D - A$ closest to a red vertex v'

flip a coin ($p = 1/M$) for it
heads: color it red
buy edge (v, v')

an incremental view



- Init: Pick a “root” r , $A = \{r\}$
- At step t
Pick vertex v in $D - A$ closest to a red vertex v'

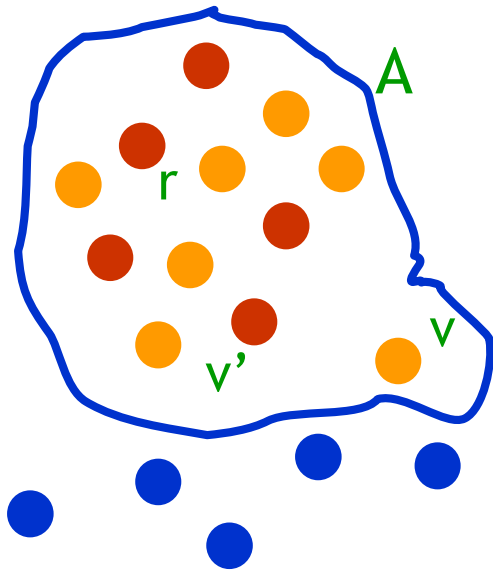
flip a coin ($p = 1/M$) for it
heads: color it red

buy edge (v, v')

tails: color it yellow

rent edge (v, v')

an incremental view



- Init: Pick a “root” r , $A = \{r\}$
- At step t
Pick vertex v in $D - A$ closest to a red vertex v'

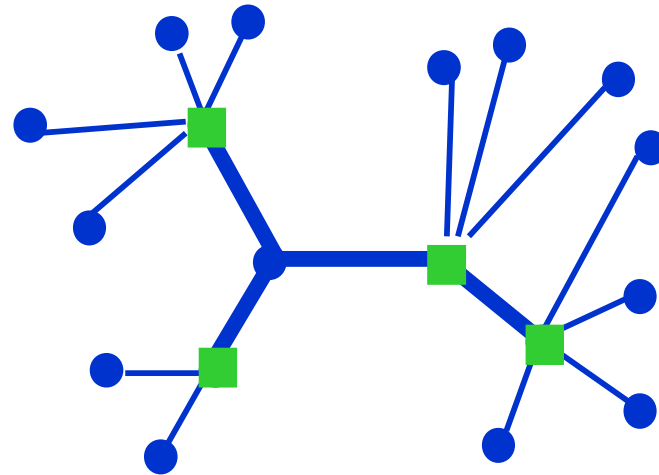
flip a coin ($p = 1/M$) for it
heads: color it red
buy edge (v, v')
tails: color it yellow
rent edge (v, v')
add v to A

note

- each vertex marked with same probability in both algorithms.
- set of bought edges = MST on F (i.e., red vertices) (Prim's algorithm)

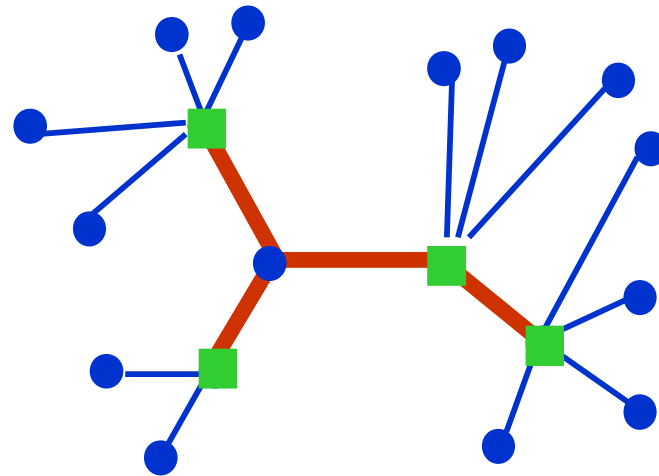
MST cost is ok

- Take OPT's tree:
 - buy her Steiner tree



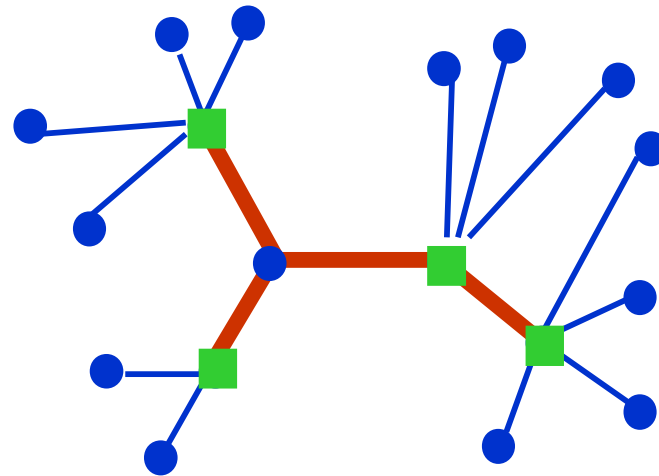
MST cost is ok

- Take OPT's tree:
 - buy her Steiner tree



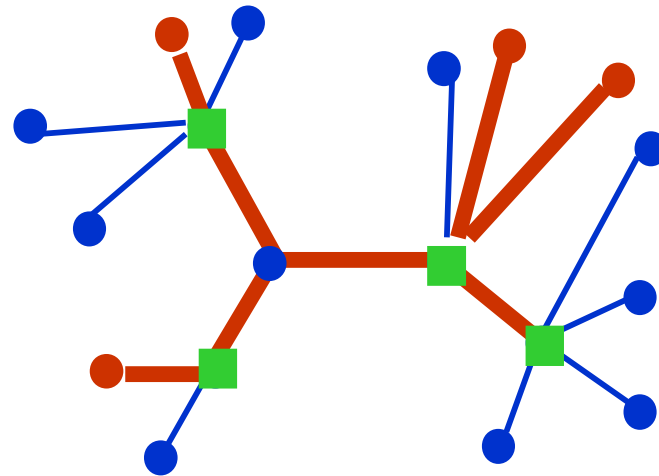
MST cost is ok

- Take OPT's tree:
 - buy her Steiner tree
 - buy paths from vertex in F to the facility



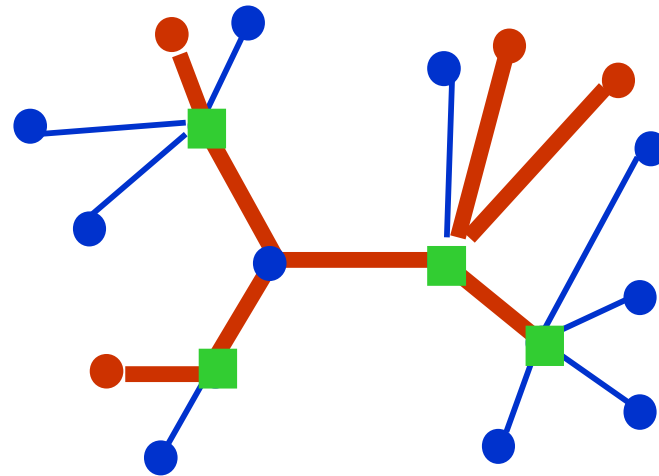
MST cost is ok

- Take OPT's tree:
 - buy her Steiner tree
 - buy paths from vertex in F to the facility



MST cost is ok

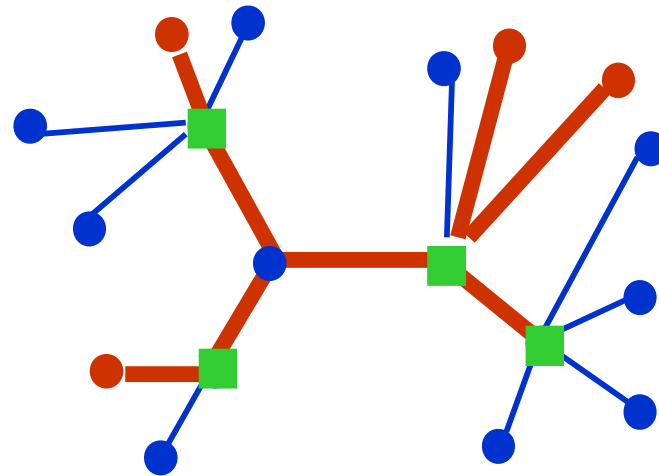
- Take OPT's tree:
 - buy her Steiner tree
 - buy paths from vertex in F to the facility
- $E[\text{Cost of this tree}]$
= Steiner(OPT) +
 $(1/M) \sum_j d(i^*_j, j)$
= OPT



MST cost is ok

- Take OPT's tree:
 - buy her Steiner tree
 - buy paths from vertex in F to the facility
- $E[\text{Cost of this tree}]$
= Steiner(OPT) +
 $(1/M) \sum_j d(i^*_j, j)$
= OPT

$\Rightarrow \text{MST} \leq 2 \text{OPT}$



what about connection cost?

- When looking at a vertex v , condition on the history H .
- $E[\text{renting cost for } v \mid H] = (1 - 1/M) \times d(v, v') < d(v, v')$
- $E[\text{buying cost for } v \mid H] = 1/M \times M d(v, v') = d(v, v')$
- $E[\text{renting cost for } v \mid H] < E[\text{buying cost for } v \mid H]$
- $E[\text{renting cost for } v] < E[\text{buying cost for } v]$
- $E[\text{total renting cost}] < E[\text{total buying cost}] \leq 2 \text{ OPT}$

$$E[\text{total cost}] \leq 4 \text{ OPT}$$

algorithm for OPT(graph)

Suppose

- $M = \text{\#senders} \leq \text{\#receivers}$

Algorithm

1. Mark a random sender
Mark each receiver (independently) with prob $1/M$
2. Build a Steiner tree on marked vertices
3. Connect everyone else to closest marked vertex

open problem

- Terminal v has a single bound $b(v)$
- Also, upper bounds $D^*(u,v)$ on $D(u,v)$

D is valid \Leftrightarrow

$$\sum_v D(u, v) \leq b(u)$$

$$D(u,v) \leq D^*(u,v)$$

- NP-hard; $O(\log n)$ approx using tree-embeddings
Can we get constant approx?
- What if we add other “cut” constraints?

open problem #3

- Can we derandomize these algorithms?

another open problem

- ConnFL is a “rent-or-buy” problem

You either buy the edges for cost M (and get unlimited capacity)
or rent them for cost 1 (and get unit capacity)

Can we solve other “rent-or-buy” problems?

Multicommodity Rent-or-Buy

primal-dual [Kumar G. Roughgarden FOCS '02]

coin-flipping [G. Kumar Pál Roughgarden]

yet another open problem

- Capacitated case?

even checking feasibility is hard
hence no approximations possible

- Can we get bicriteria approximations
where we violate capacities by a constant
and get within constants of OPT?

open problem #49

- Is this the correct cost measure?

E.g., for optical networks:

- minimize congestion
(i.e., the number of fibers that have to be used)
- cost of switches at different nodes
- non-linear costs for edges
- allow some rearrangements (instead of non-blocking)

see, e.g., [Saengudomlert Modiano Gallager INFOCOM '03]

that's all, folks!
