# Multicommodity Rent or Buy: Approximation Via Cost Sharing

Martin Pál

Joint work with

Anupam Gupta    Amit Kumar

Tim Roughgarden

# Talk Outline

ïCost sharing algorithm for Steiner forest:
what do we want

ïCost sharing for Steiner forest +
Simple randomized analysis =
Approximation for Rent or Buy

ïHow does the cost sharing work

# Steiner Forest and Rent or Buy

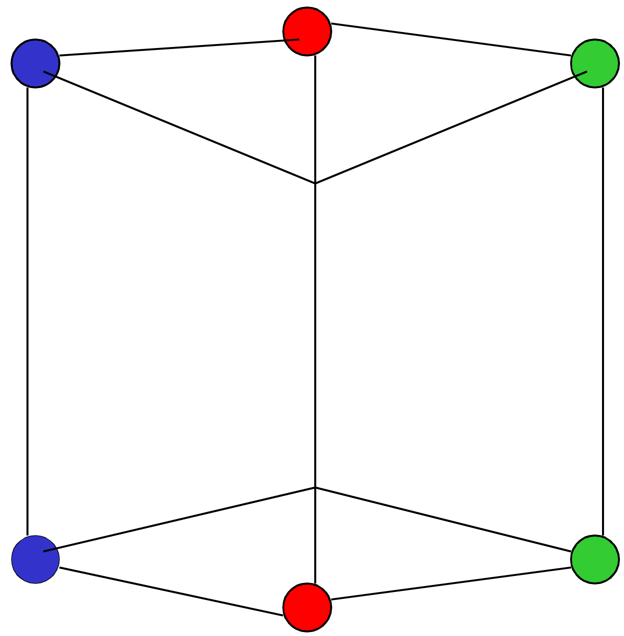
Input:

ïWeighted graph G

ïSet of demand pairs D

Solution:

A set of paths, one for each $(s_i,t_i)$ pair

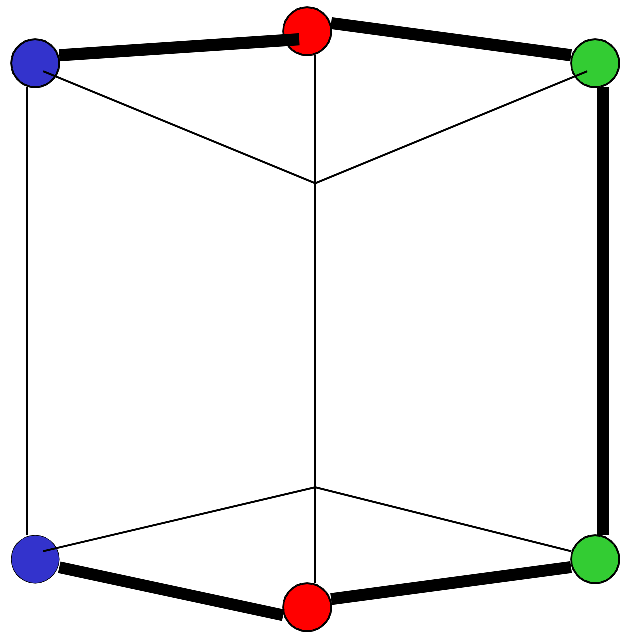


# Steiner Forest and Rent or Buy

Input:

ïWeighted graph G

ïSet of demand pairs D



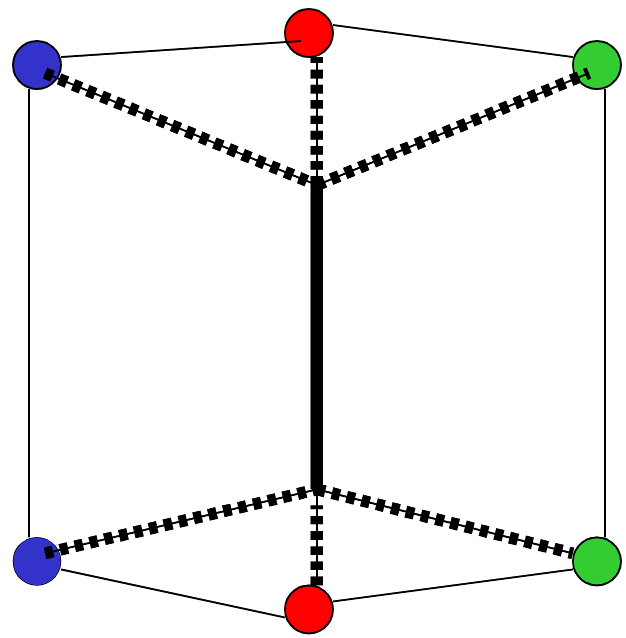
Steiner Forest: pay $c_e$ for each edge e we use, regardless of how many paths use it

# Steiner Forest and Rent or Buy

Input:

ïWeighted graph G

ïSet of demand pairs D



Rent or Buy: pay $c_e$ per pair for renting, or buy for $M \cdot c_e$

# Related Work

ï2-approximation for Steiner Forest
[Agarwal, Klein, Ravi 91] , [Goemans, Williamson 95]

ïConstant approximation for Rent or Buy
[Kumar, Gupta, Roughgarden 02]

ïSpanning Tree + Randomization = Single Sink Rent or Buy
[Gupta, Kumar, Roughgarden 03]

# Cost Sharing for Steiner Forest

Cost sharing algorithm: approximation algorithm that given $D$, computes solution $F_D$ and set of cost shares $\xi(D,j)$ for $j \in D$

# Cost Sharing for Steiner Forest

Cost sharing algorithm: approximation algorithm that given $D$, computes solution $F_D$ and set of cost shares $\xi(D,j)$ for $j \in D$

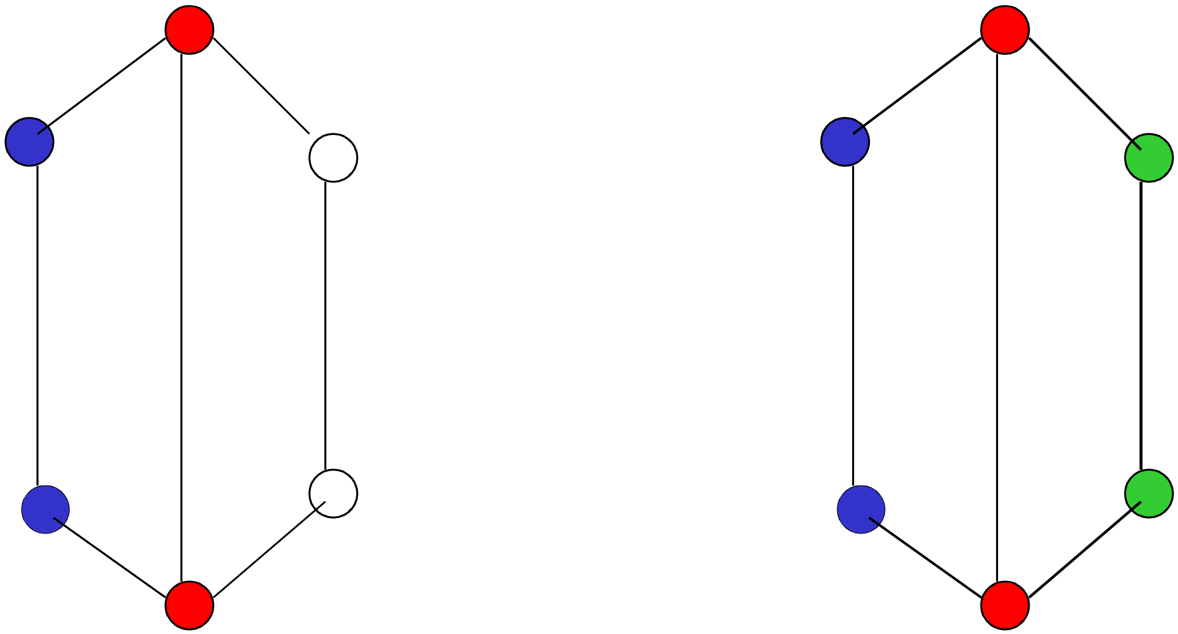
(P1) Constant approximation: $\mathrm{cost}(F_D) \leq \alpha\, St^*(D)$

(P2) Cost shares do not overpay: $\sum_j \xi(D,j) \leq St^*(D)$

(P3) Cost shares pay enough: let $D' = D \cup \{s_i, t_i\}$

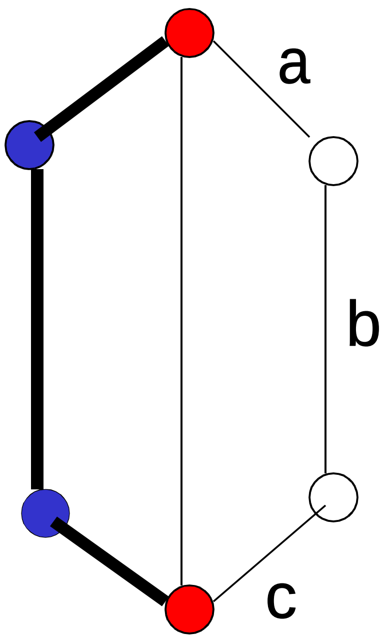
dist($s_i$, $t_i$) in $G/\ F_D \leq \beta\, \xi(D',i)$

# What does (P3) say

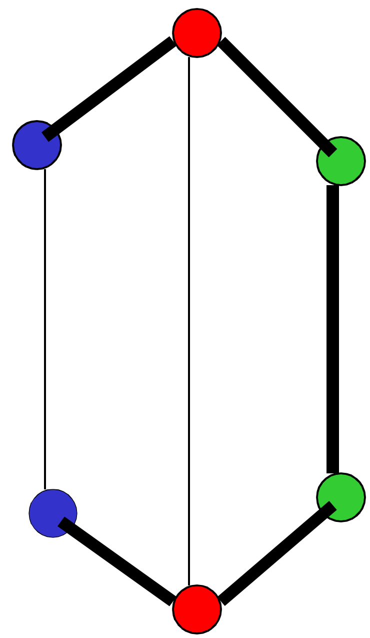


# What does (P3) say

Solution for




Solution for



$\xi(\;\;,\;\;) \geq \beta\,(a+c)$

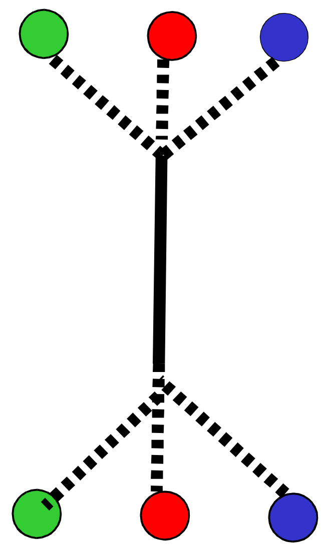
# The Rent or Buy Algorithm

Algorithm SimpleMRoB:

3. Mark each demand pair independently at random with prob. $1/M$. Let $S$ be the marked set.
4. Use a cost sharing algorithm to build a Steiner forest on the marked set $S$.
5. Rent shortest paths between all unmarked pairs.

# Analysis

Claim: Expected cost of buying an optimal Steiner forest on S is at most OPT.



Corollary: Expected cost of step 2 $\leq \alpha \cdot$ OPT

Corollary: Expected sum of cost shares $\leq$ OPT

# Analysis (2)

Claim: Expected cost of step 3 is at most $\beta \cdot OPT$.

Let $S' = S - \{i\}$

$E[\text{cost share of } i \mid S'] = 1/M \cdot M\, \xi(S'+i, i)$

$E[\text{rental cost of } i \mid S'] = (M-1)/M \cdot (\text{dist}(s_j, t_j) \text{ in } G/F_{S'})$

From (P3):

$E[\text{rental cost of } i \mid S'] \leq \beta \cdot E[\text{cost share of } i \mid S']$

$E[\text{rental cost of } i] \leq \beta \cdot E[\text{cost share of } i]$

# Analysis (3)

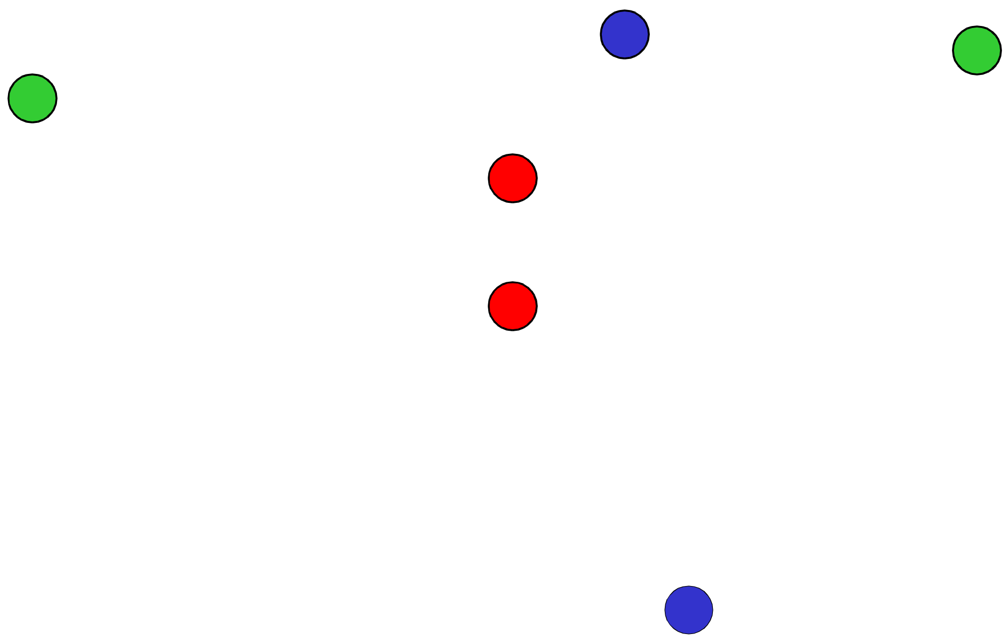
There is a cost sharing algorithm with $\alpha = 6$, $\beta = 6$.

Theorem: SimpleMRoB is a 12-approximation algorithm.

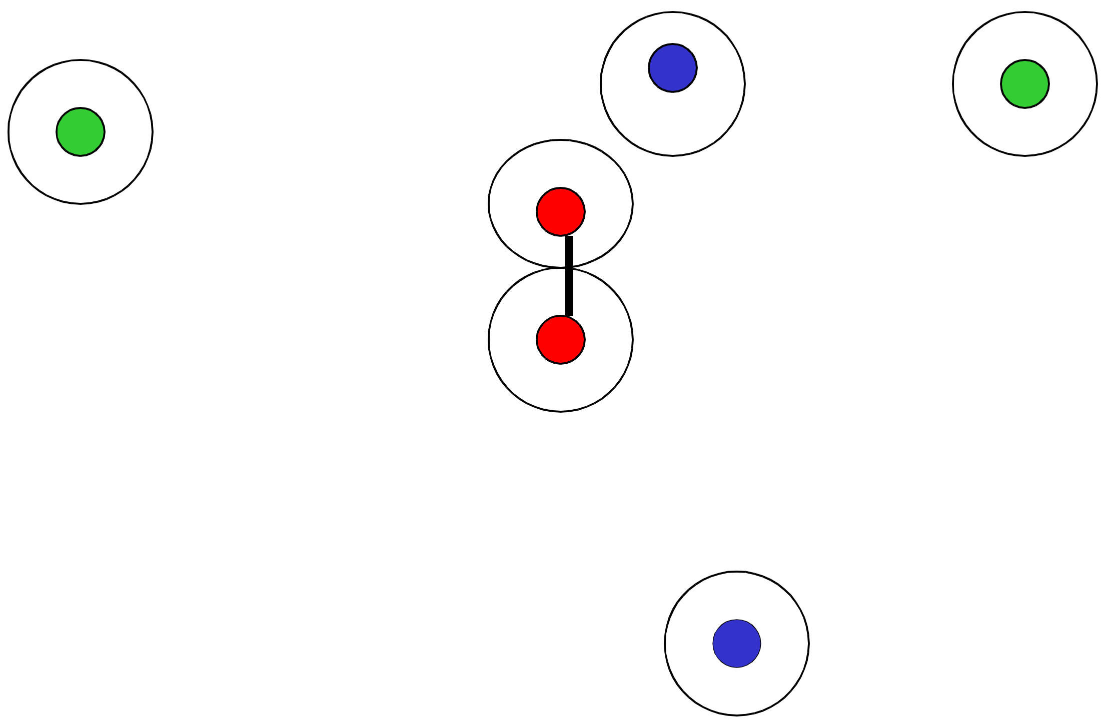
# Existing Steiner forest algorithms

- Each demand starts in a separate cluster
- Active clusters grow at unit rate
- When two clusters touch, they merge into one
- Demands may get deactivated
- A cluster is deactivated if it has no active demands
- Inactive clusters do not grow
- Keep adding enough edges so that all active demands in a component are connected

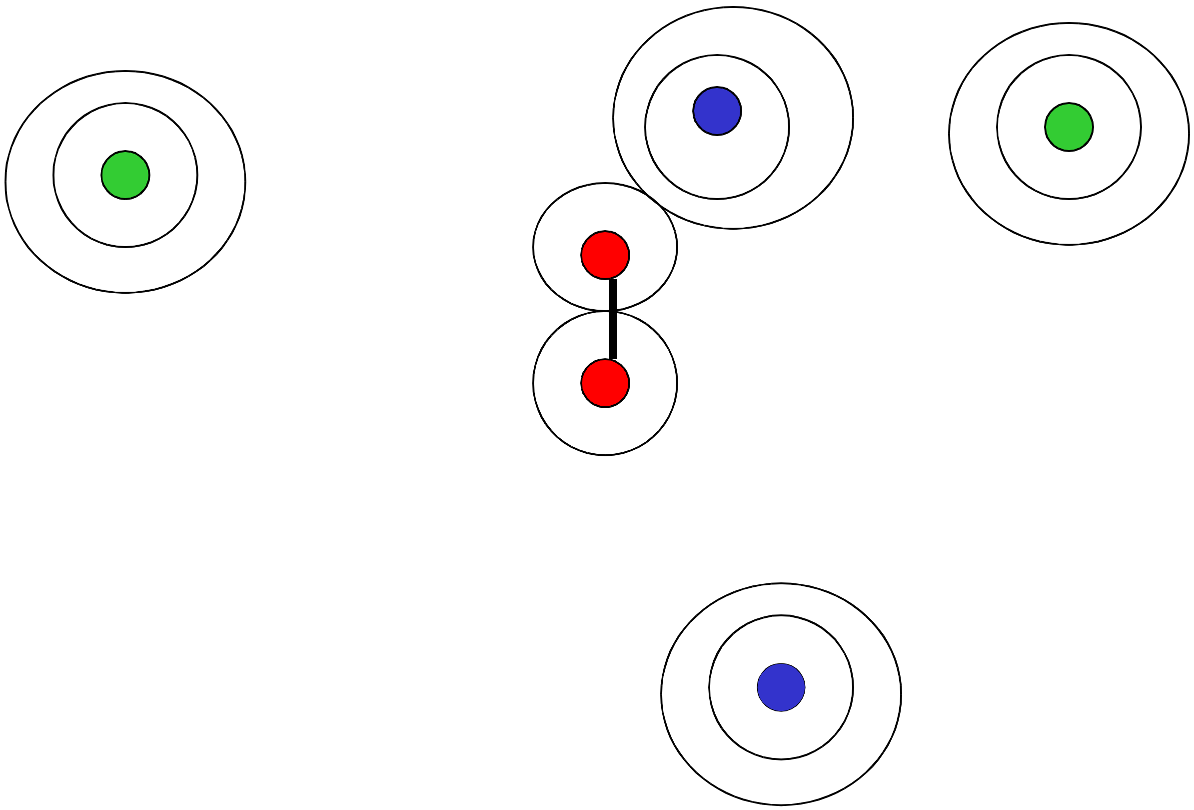
# Existing Steiner forest algorithms



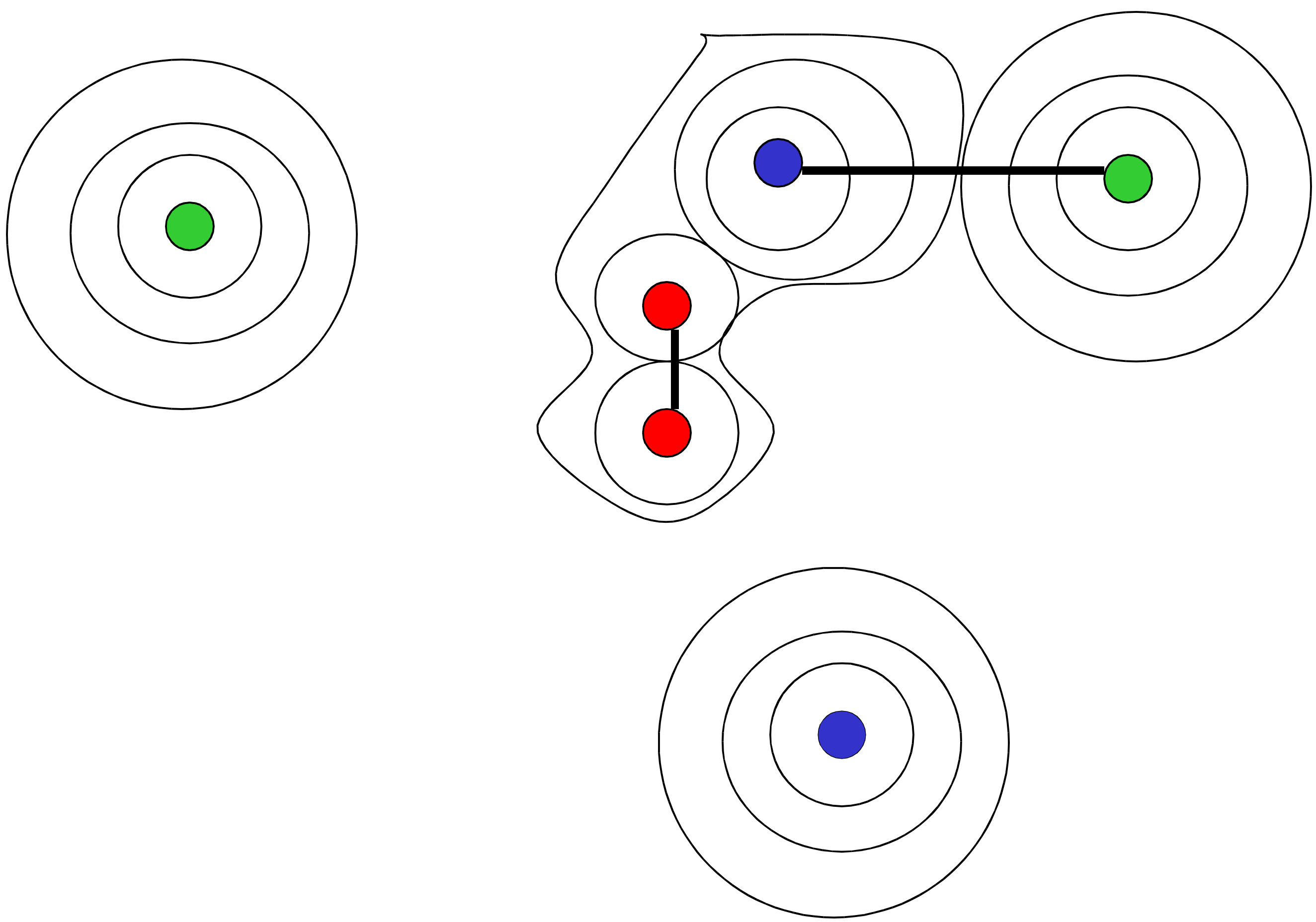
# Existing Steiner forest algorithms



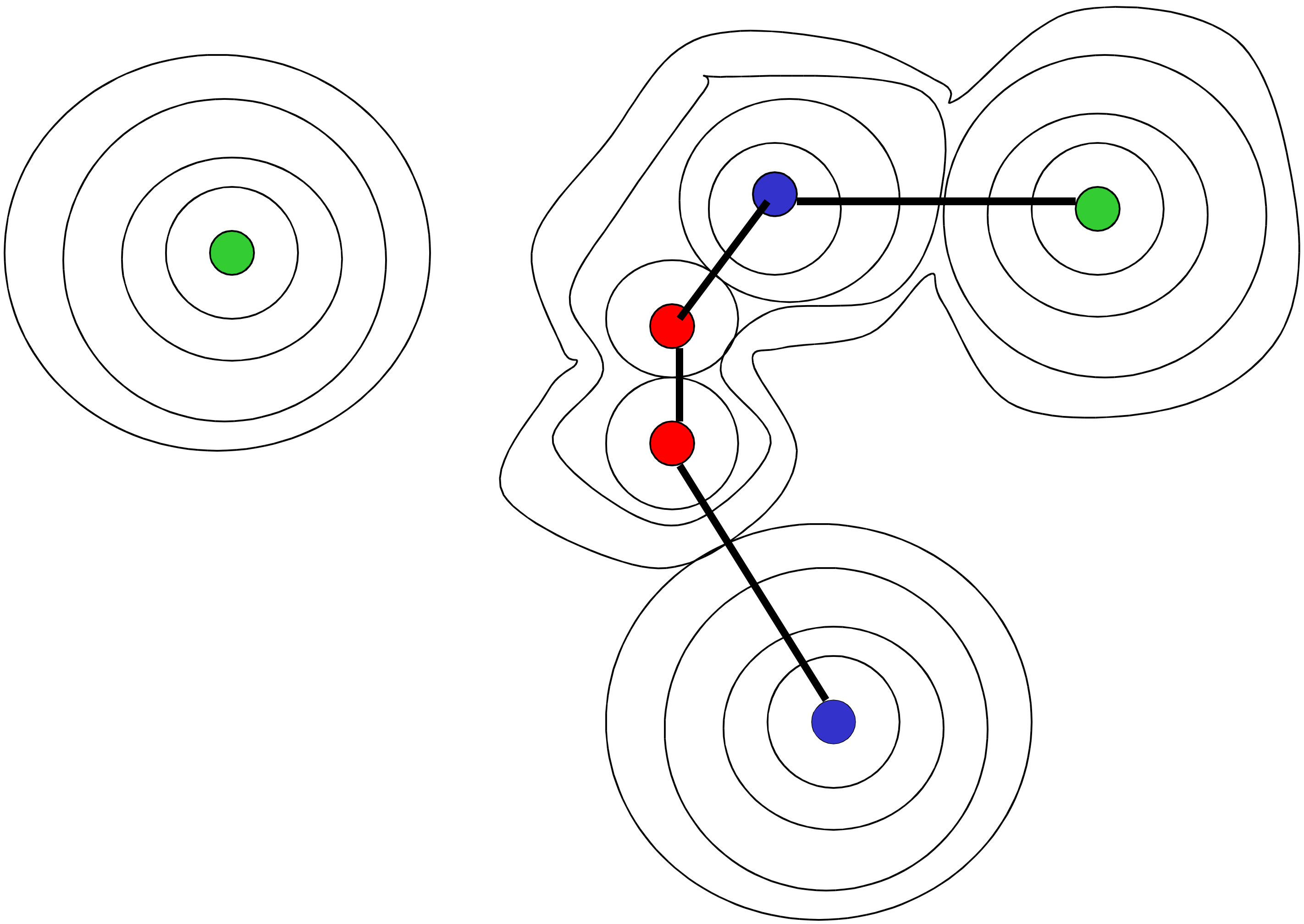
# Existing Steiner forest algorithms



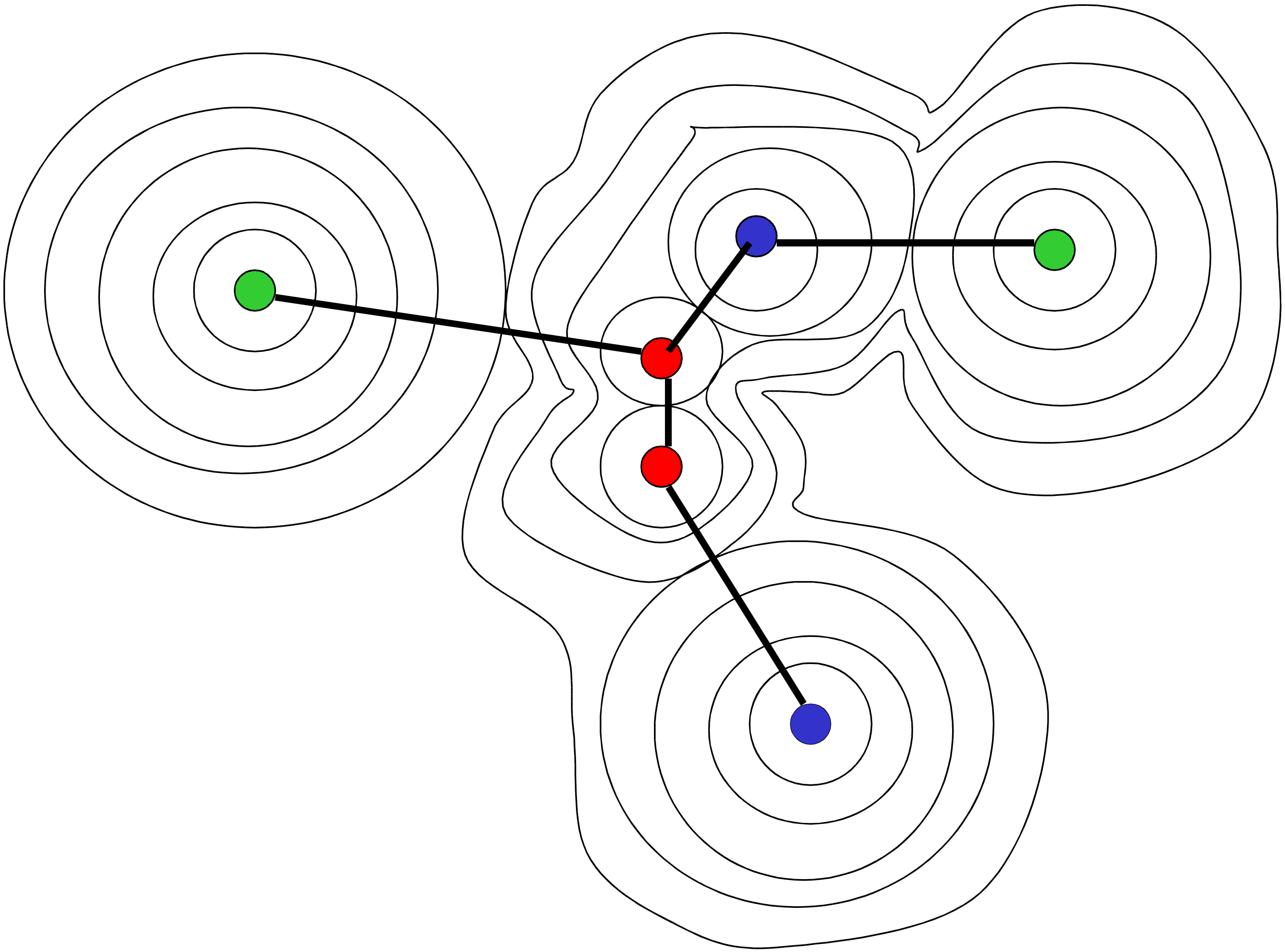
# Existing Steiner forest algorithms



# Existing Steiner forest algorithms



# Existing Steiner forest algorithms



# How to define cost shares

Need to pay for the growth of the clusters

Active demands share equally the cost of growth of a cluster.

$a(u,\tau)$ = $1$ / ($\#$ of active demands in cluster with $u$)

= $0$ if $u$ not active at time $\tau$

cost share of $u = \int a(u,\tau) \, d\tau$

# Does it work?

(P1) Constant approximation: $\text{cost}(F_D) \leq \alpha\ St^*(D)$

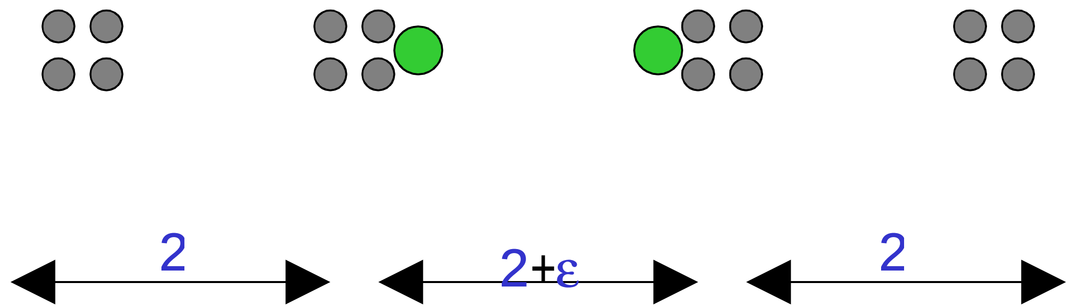
(P2) Cost shares do not overpay: $\sum_j \xi(D,j) \leq St^*(D)$

(P3) Cost shares pay enough: let $D' = D \cup \{s_i,\ t_i\}$

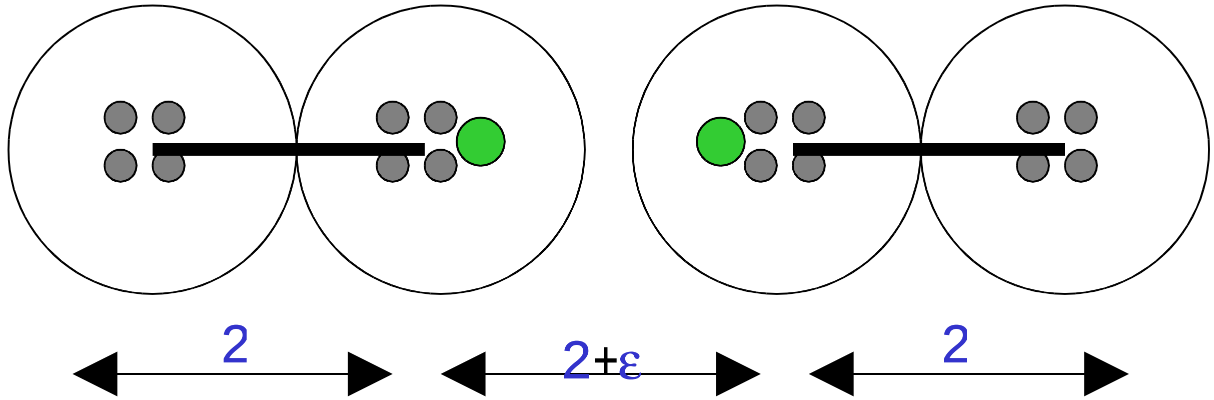
$\text{dist}(s_i,\ t_i)$ in $G/\ F_D \ \leq\ \beta\ \xi(D',i)$

(P1) and (P2) easy to verify. How about (P3)?

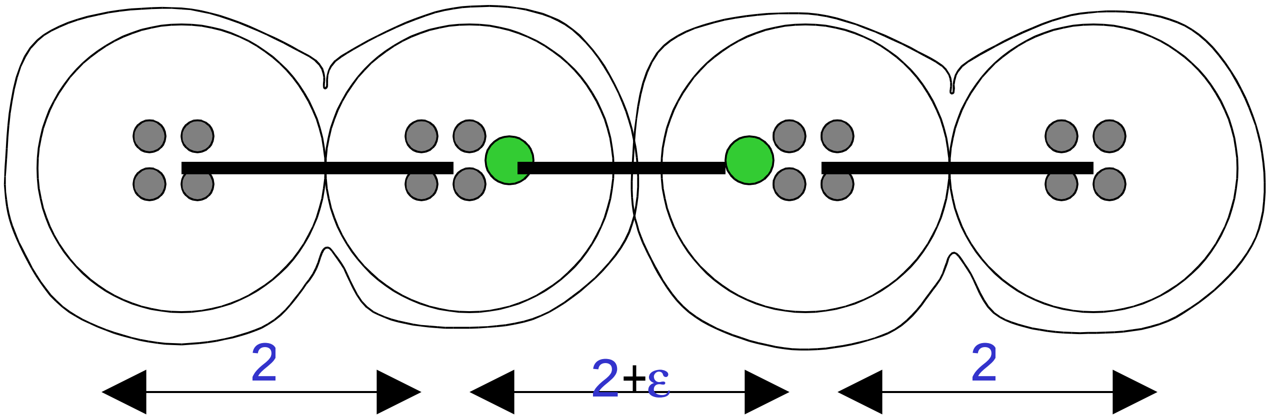
# Does it work?

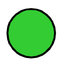


# Does it work?

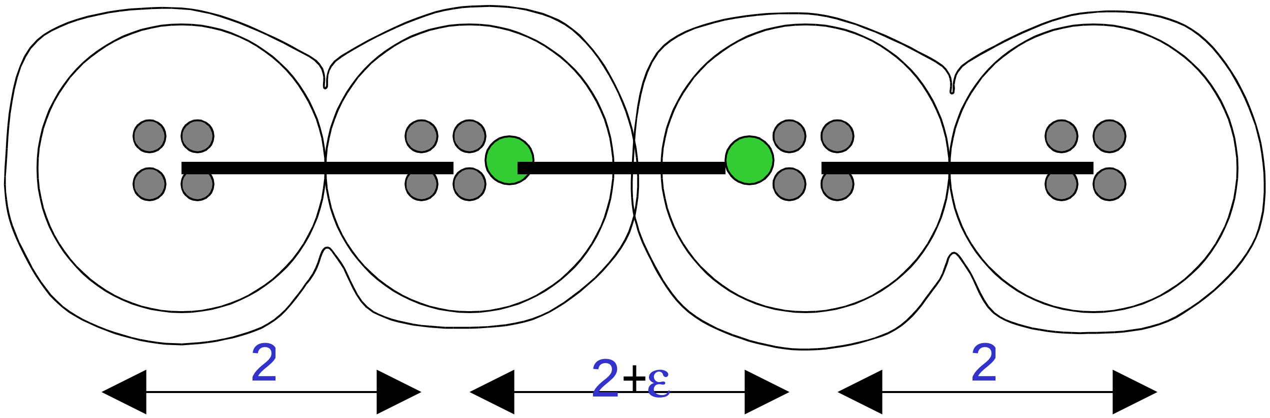


# Does it work?

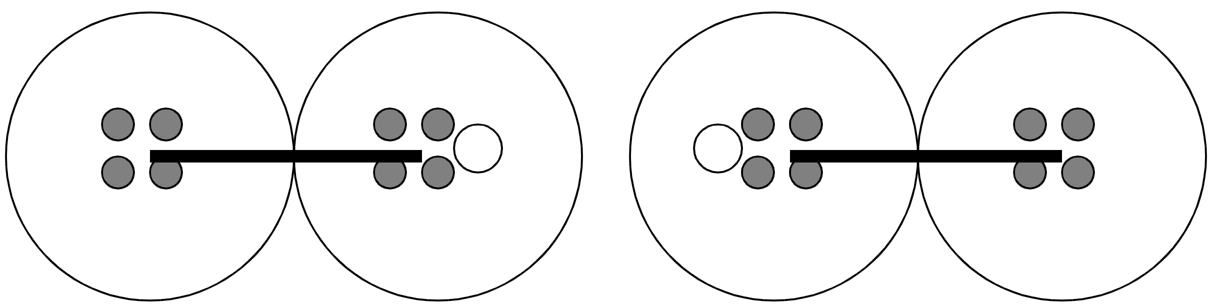



cost share of  = $2/n + \varepsilon$

# Does it work?




cost share of = $2/n + \varepsilon \ll 2 + \varepsilon$



solution without

# Solution

Inflate the balls !

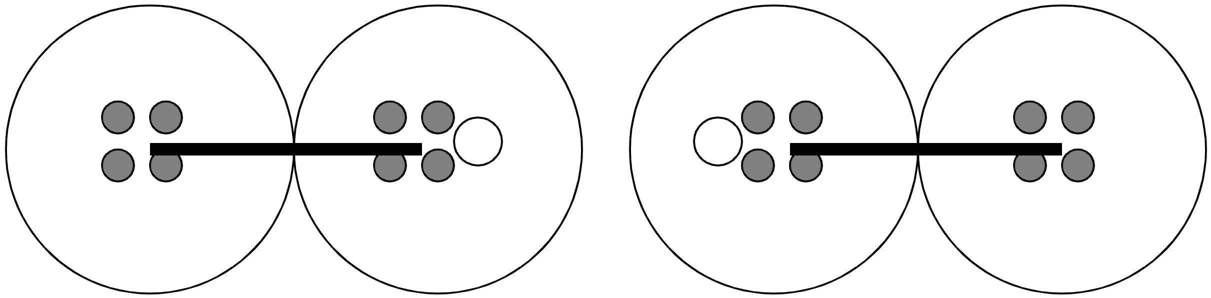
3. Run the standard [AKR, GW] algorithm
4. Note the time $T_j$ when each demand j deactivated
5. Run the algorithm again, except that now every demand j deactivated at time $\gamma \cdot T_j$ for some $\gamma > 1$

Adopting proof from [GW]: buying cost at most $2 \cdot \gamma \cdot$OPT.

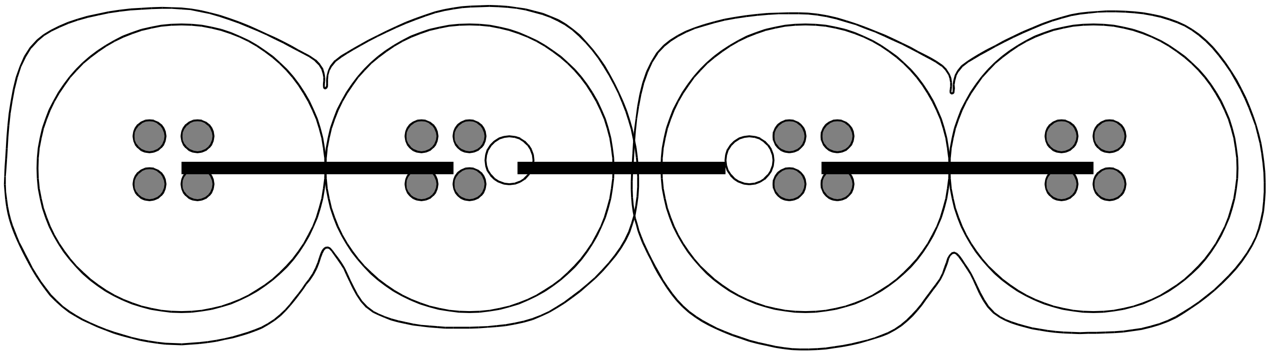
# Works on our example

$\gamma = 2$

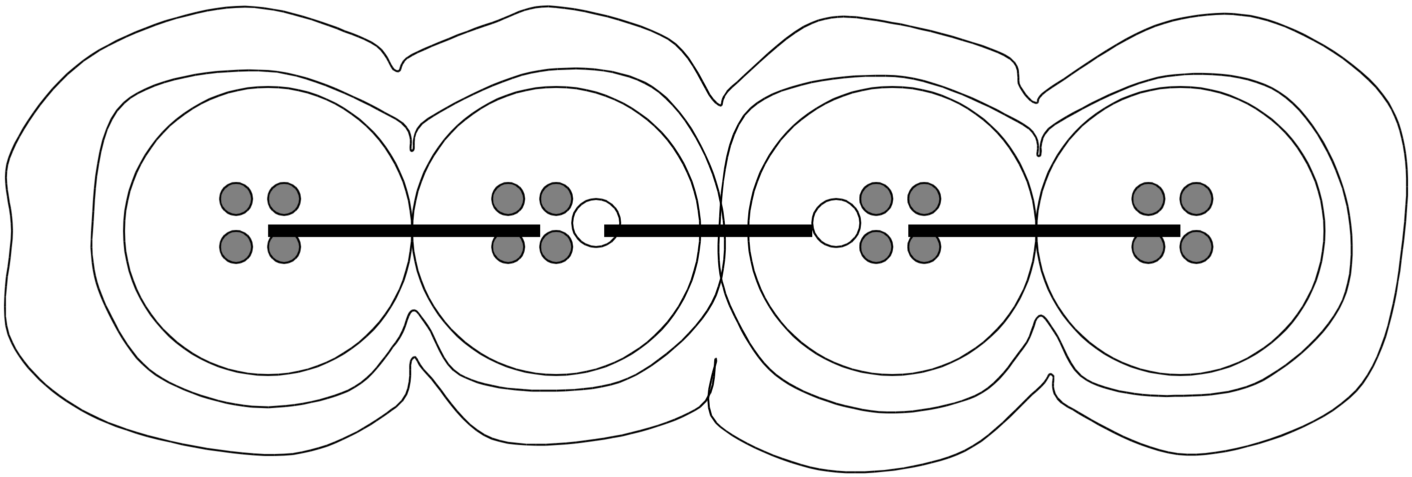
# Works on our example



# Works on our example



# Works on our example



# How to prove (P3)

Need to compare runs on $D$ and $D' = D \cup \{s_i, t_i\}$

Idea: 1. pick a $\{s_i, t_i\}$ path $P$ in $G / F_D$

2. Show that $\xi(D',i)$ accounts for a constant fraction of length($P$)

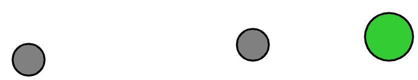
# How to prove (P3)

Need to compare runs on $D$ and $D' = D \cup \{s_i, t_i\}$

Idea: 1. pick a $\{s_i, t_i\}$ path $P$ in $G/F_D$

2. Show that $\xi(D',i)$ accounts for a constant fraction of length($P$)

Instead of $\xi(D',i)$, use alone($i$), the total time $s_i$ or $t_i$ was alone in its cluster

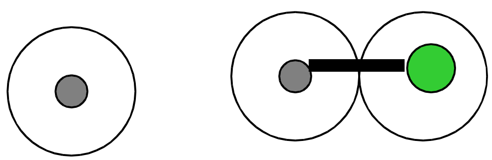


# How to prove (P3)

Need to compare runs on $D$ and $D' = D \cup \{s_i, t_i\}$

Idea: 1. pick a $\{s_i, t_i\}$ path $P$ in $G / F_D$

2. Show that $\xi(D',i)$ accounts for a constant fraction of length($P$)

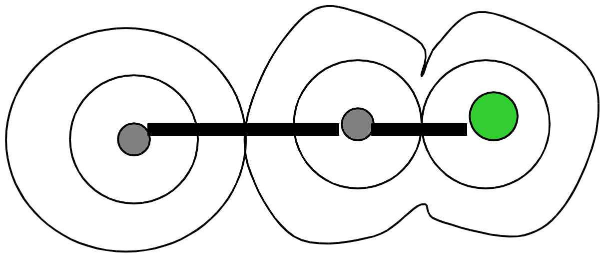


# How to prove (P3)

Need to compare runs on $D$ and $D' = D \cup \{s_i, t_i\}$

Idea: 1. pick a $\{s_i, t_i\}$ path $P$ in $G/F_D$

2. Show that $\xi(D',i)$ accounts for a constant fraction of length($P$)

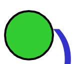


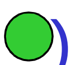
# How to prove (P3)

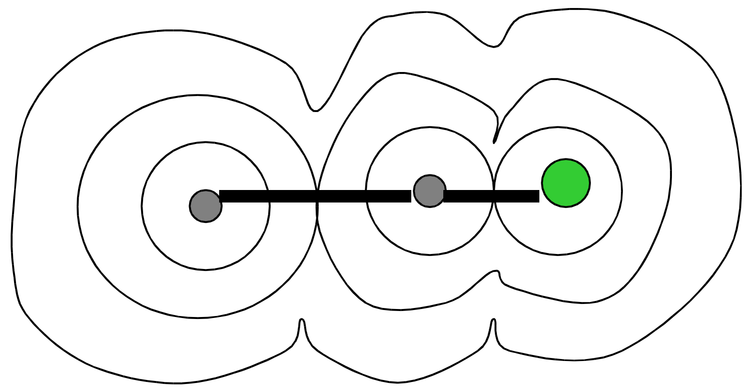
Need to compare runs on $D$ and $D' = D \cup \{s_i, t_i\}$

Idea: 1. pick a $\{s_i, t_i\}$ path $P$ in $G / F_D$

2. Show that $\xi(D',i)$ accounts for a constant fraction of length($P$)

$\xi$(D', ), = 2.5

alone( ) = 2.0



# Mapping of layers

Lonely layer: generated by $s_i$ or $t_i$ while the only active demand in its cluster

Each non-lonely layer in the $D'$ run maps to $\gamma$ layers in the scaled $D$ run.

Idea:

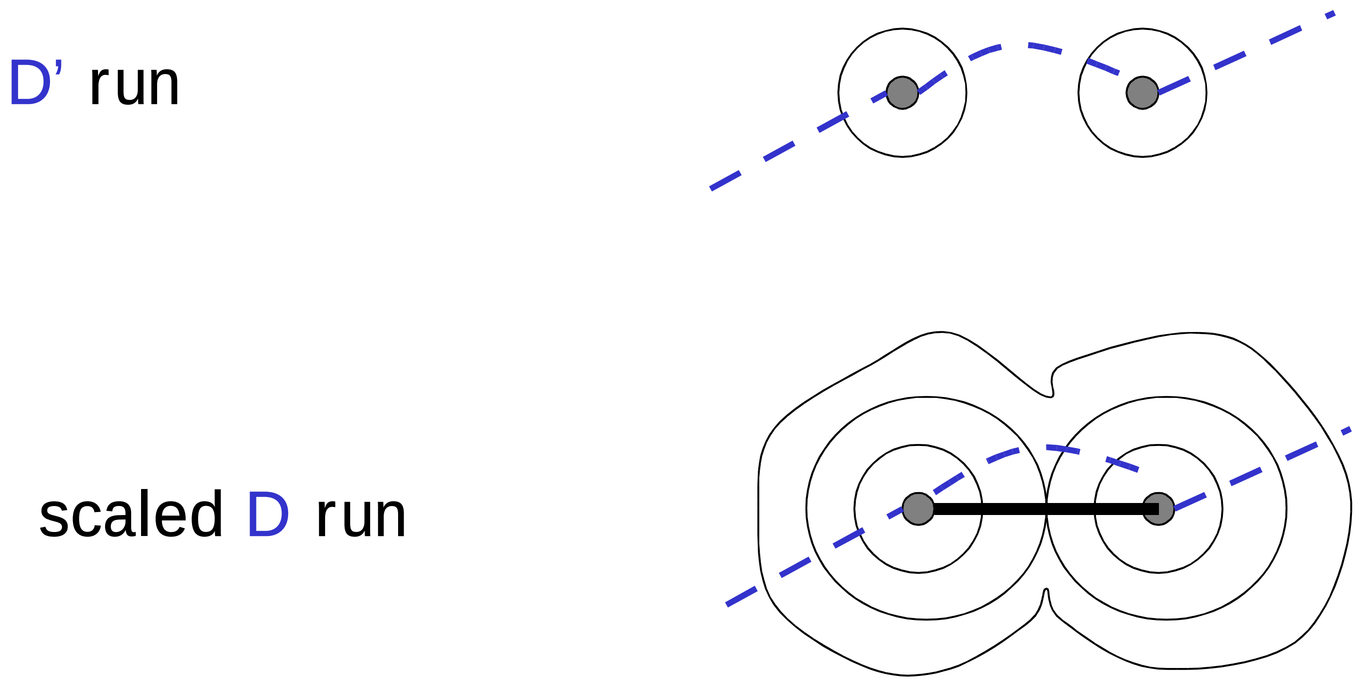
length($P$) = #of lonely and non-lonely layers crossed in $D'$ run

length($P$) $\geq$ #of layers it crosses in scaled $D$ run

Hence: for each non-lonely layer, $\gamma-1$ lonely layers crossed.

# Mapping not one to one

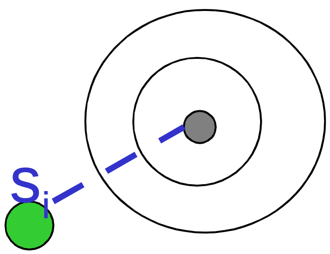
Two non-lonely layers in D' run can map to the same layer in scaled D run.



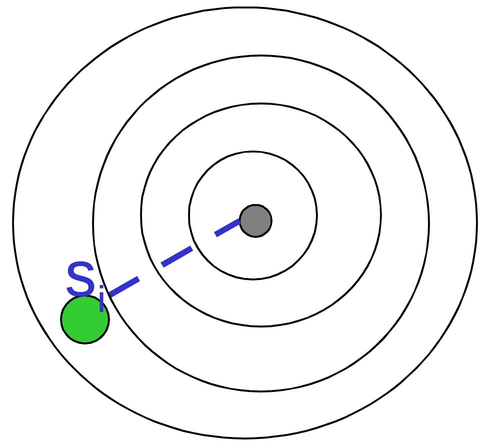
# Not all layers of D run cross P

A layer in D' run that crosses P can map to a layer in scaled D run that does not cross P

D' run




scaled D run




The "waste" can be bounded.

# Open problems

- Does SimpleMRoB work with unscaled GW? With an arbitrary Steiner forest subroutine?
- Other cost sharing functions? (crossmonotonic..)