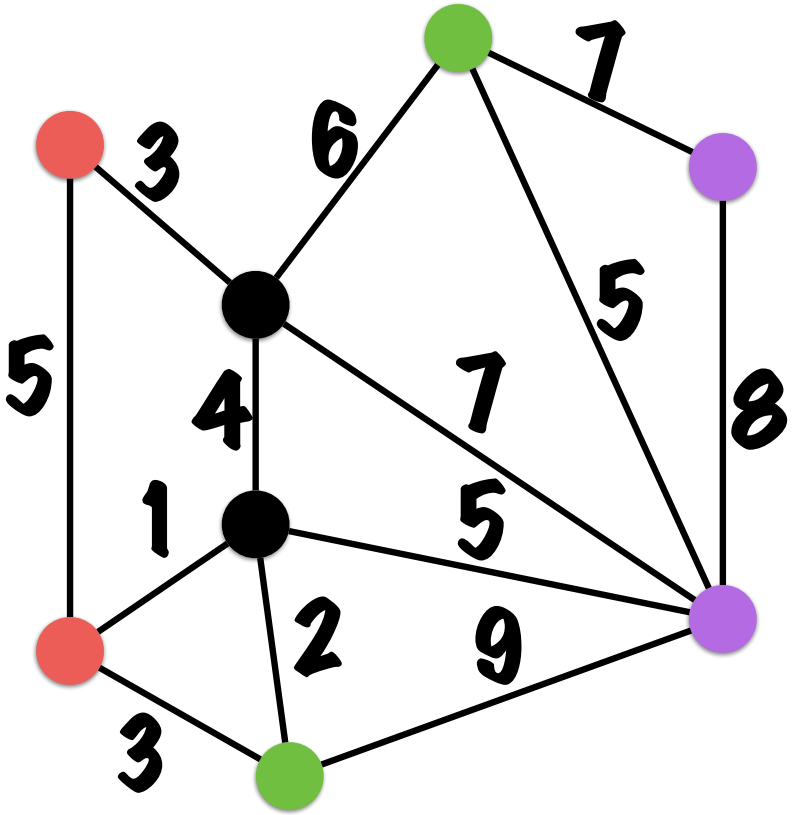
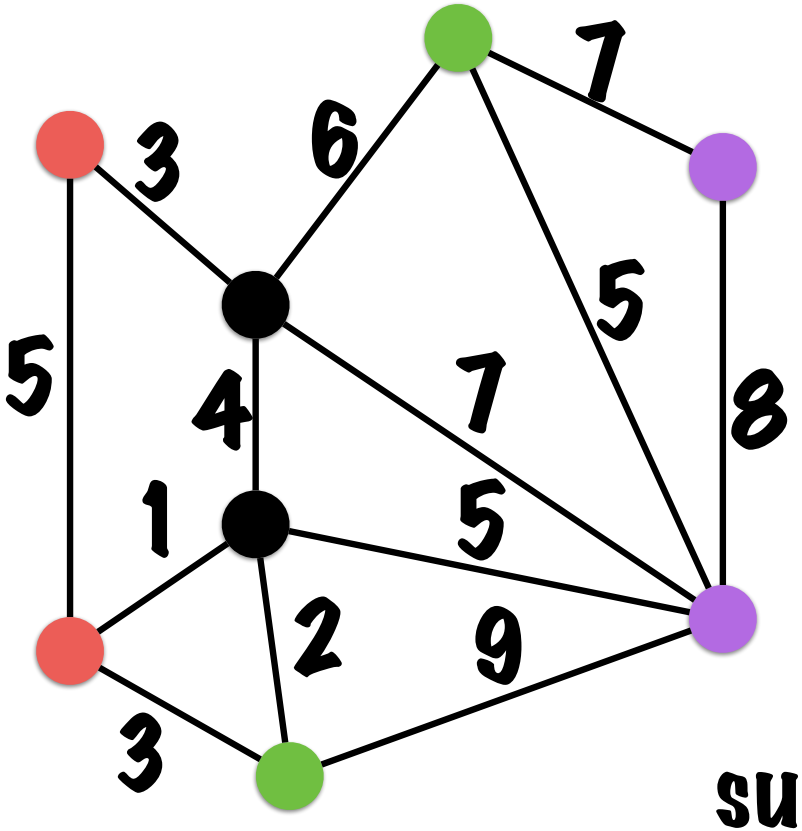
Steiner forest

Appears in...



VLSI
Optical and wireless communication
Transportation and distribution systems
Network design

Connecting subsets



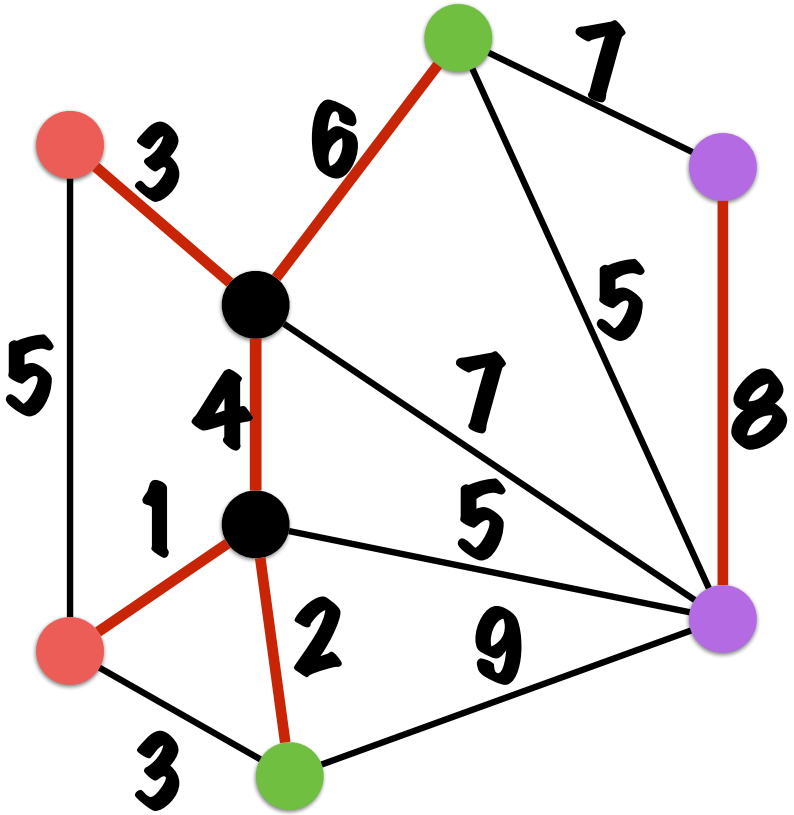
Input

Graph with edge costs

subsets of vertices (terminals)

{●, ●}, {●, ●}, {●, ●}

Connecting subsets



Input

Graph with edge costs

subsets of terminals

{ ● , ● }, { ● , ● },{ ● , ● }

Output

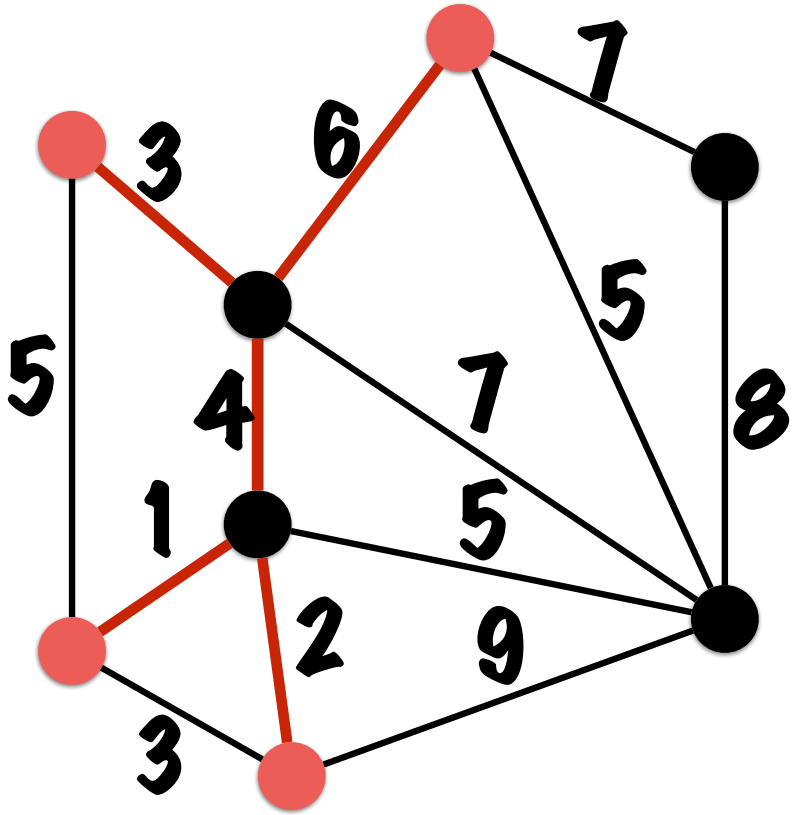
choose edges s.t.

each subset is connected

Goal

minimize cost of edges

A special case: Steiner tree



Input

Graph with edge costs

one subset of terminals

{ ●, ●, ●, ● }

Output

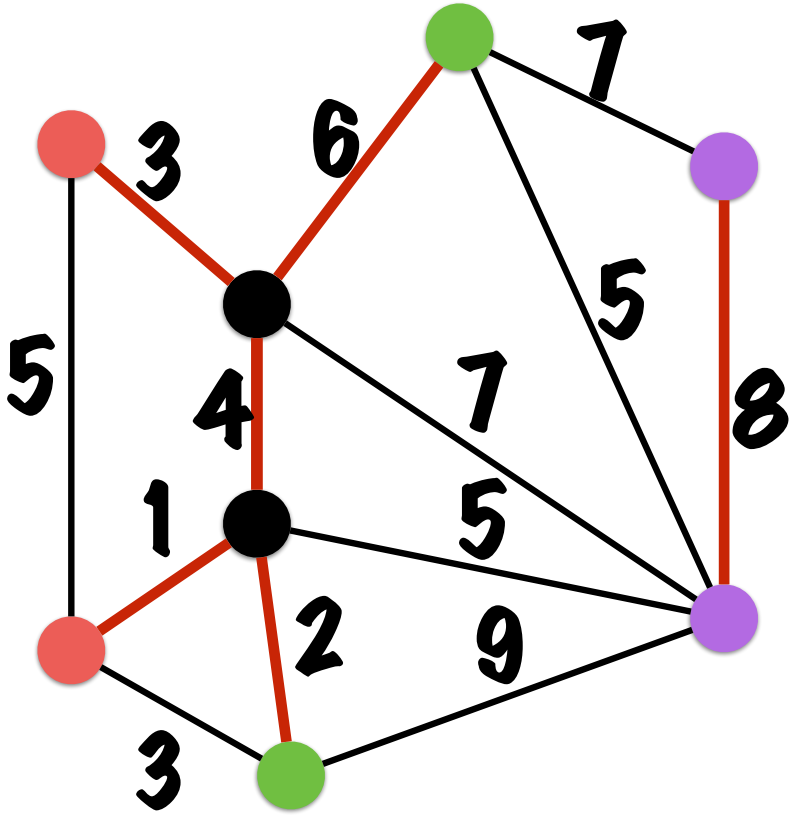
choose edges s.t.

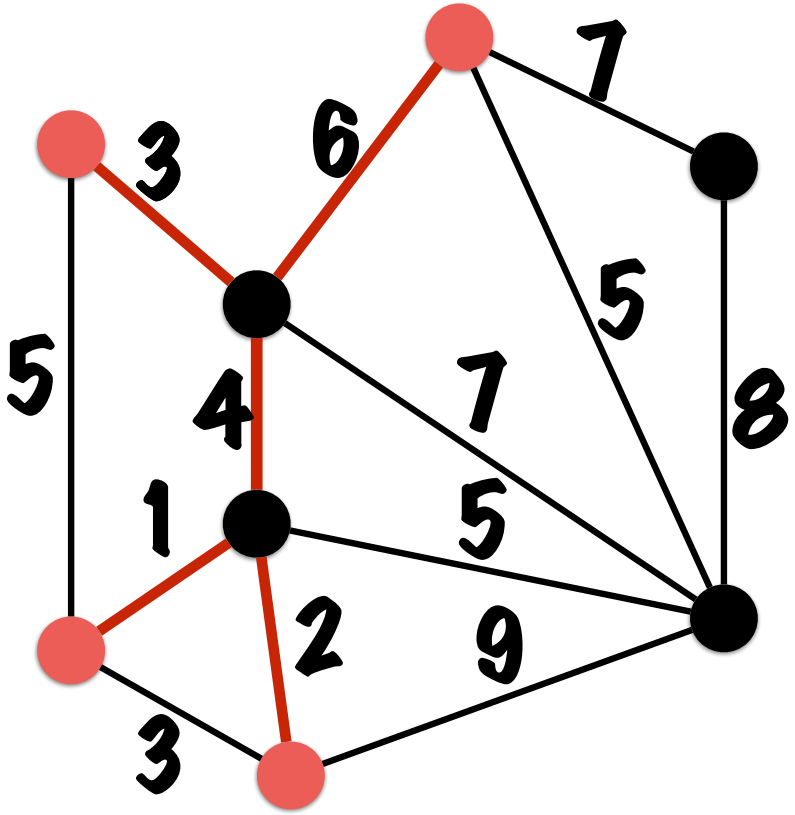
the subset is connected

Goal

minimize cost of edges

Steiner forest vs. Steiner tree





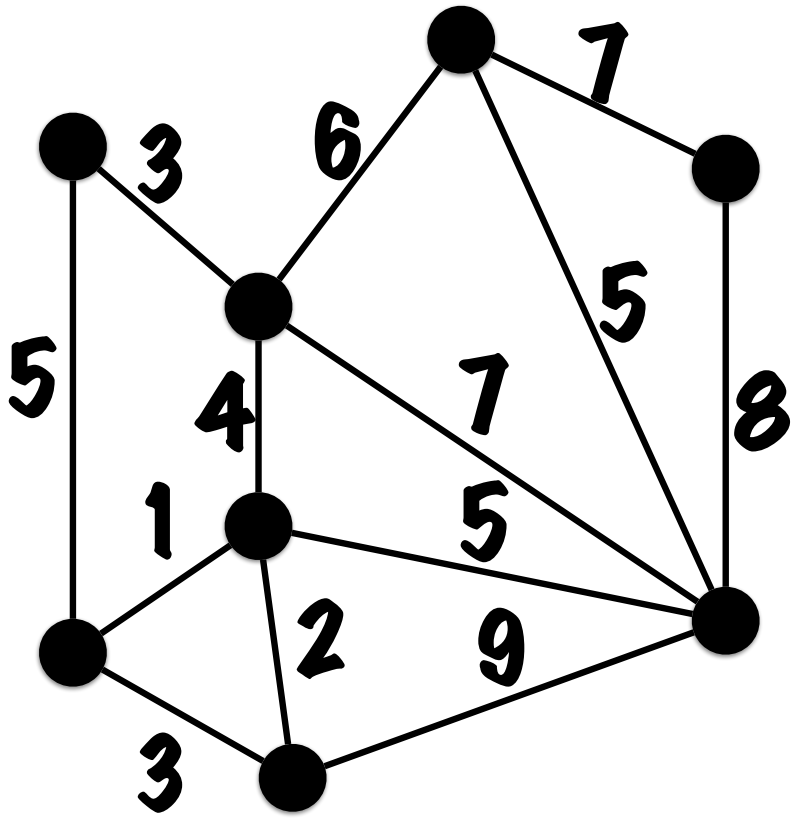
Steiner forest

Steiner forest

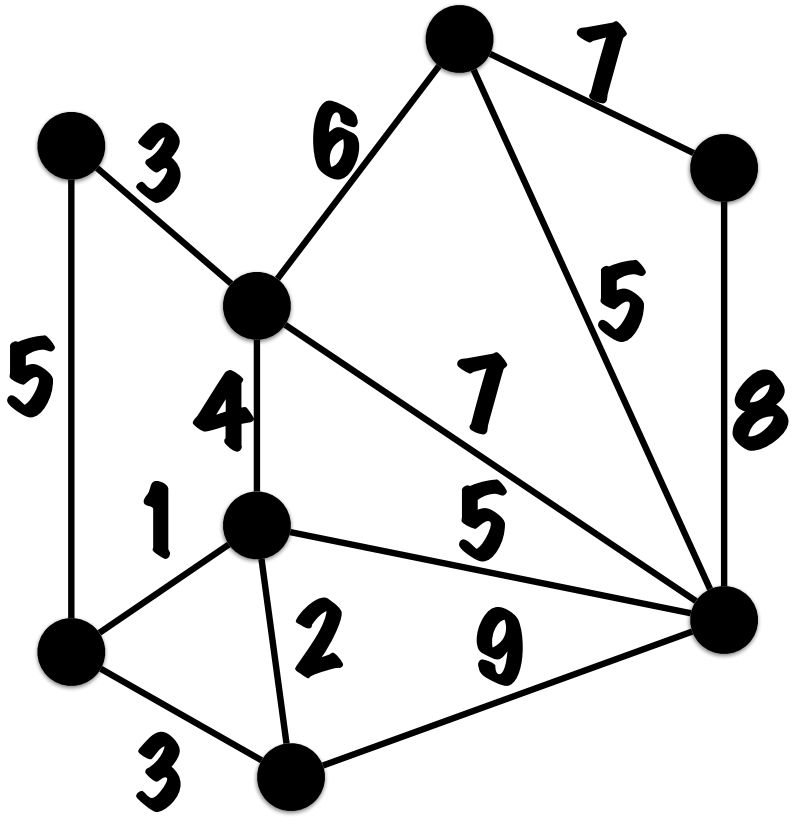
Steiner tree

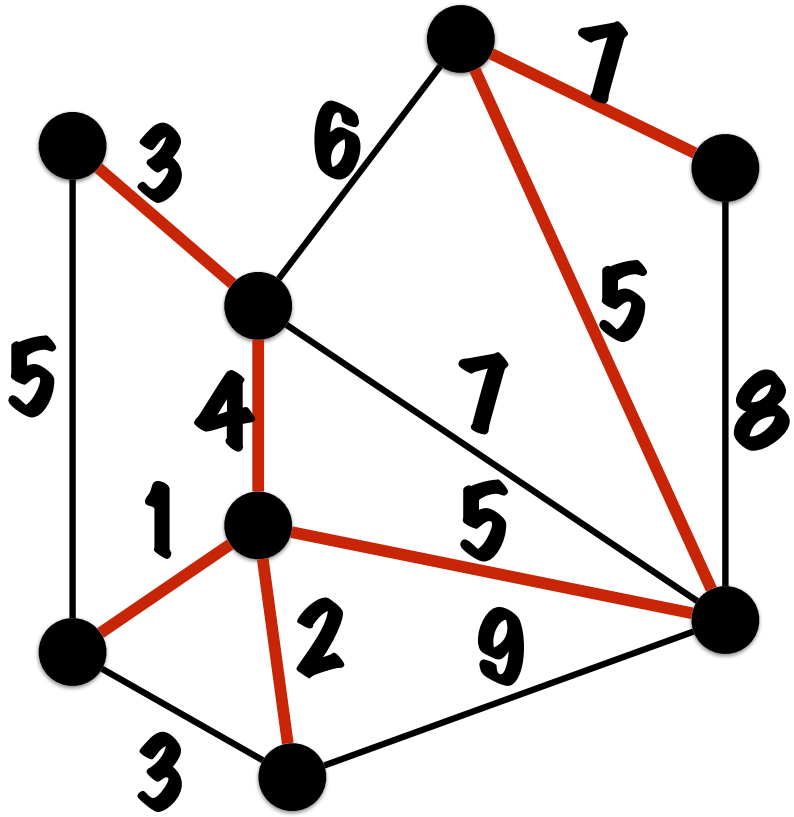
Observe:

if the subset = all nodes
then Steiner tree =
minimum spanning tree



MST

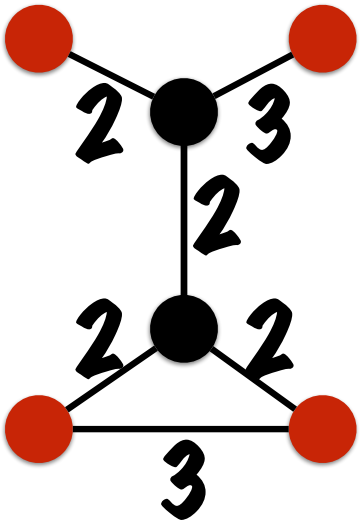


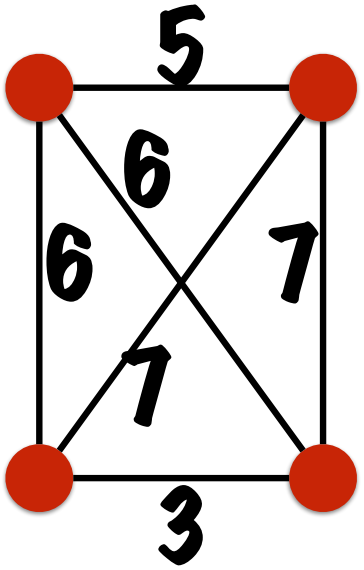


Steiner tree approx. algorithm

1. Define new graph:

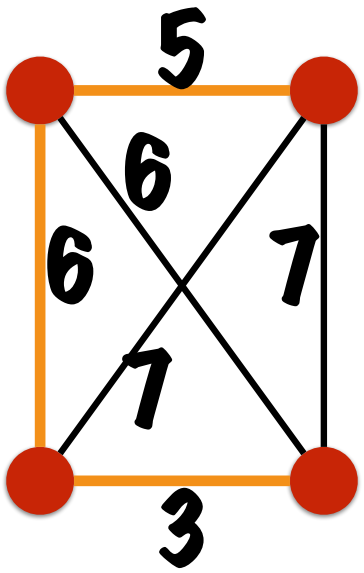
- Vertex set = S
- Edge set = complete graph
- Weight of {u,v} = shortest path length in G





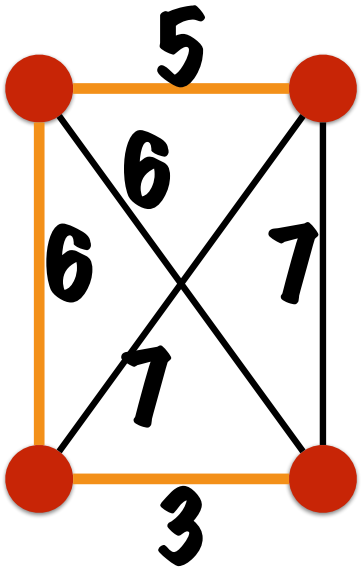
Steiner tree algorithm

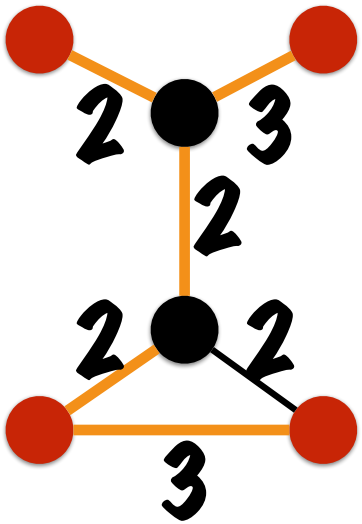
2. Compute minimum spanning tree on new graph



Steiner tree algorithm

3. Output corresponding set of original edges





Here
Output=12
OPT=11

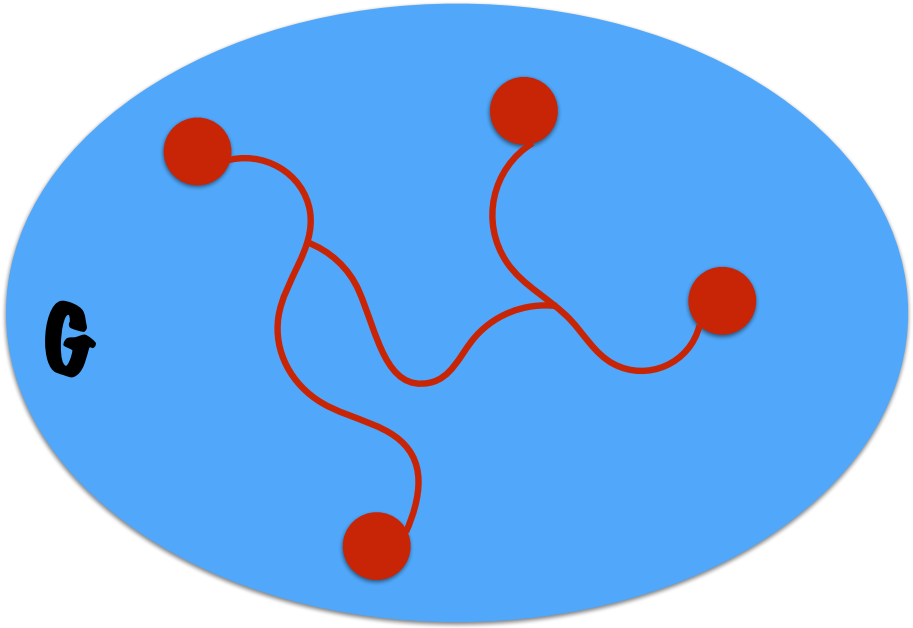
Steiner tree algorithm

1. new complete graph on subset
2. minimum spanning tree on that graph
3. output corresponding set of original edges

Theorem: it's a 2-approximation

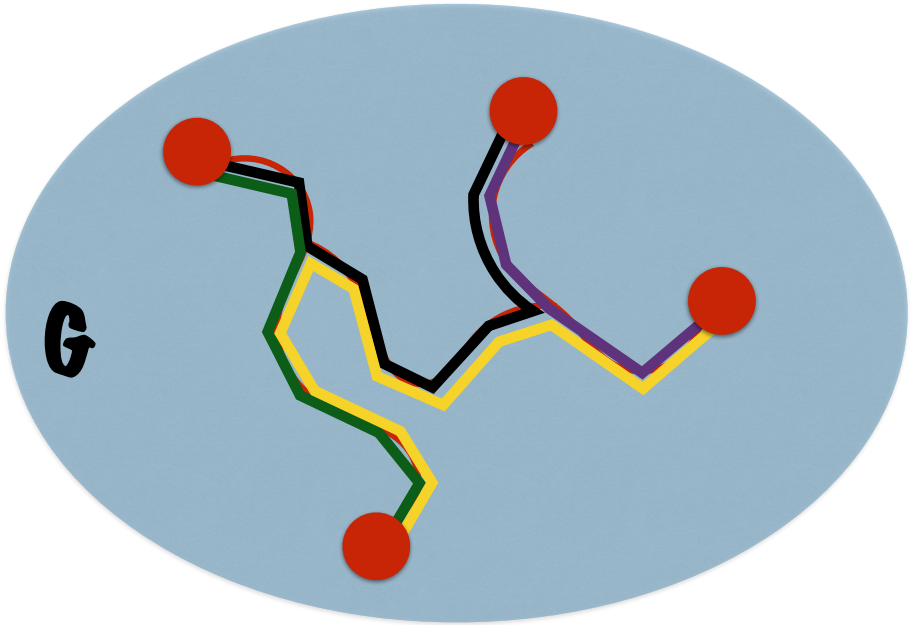
Theorem: it's a 2-approximation

Consider unknown OPT tree



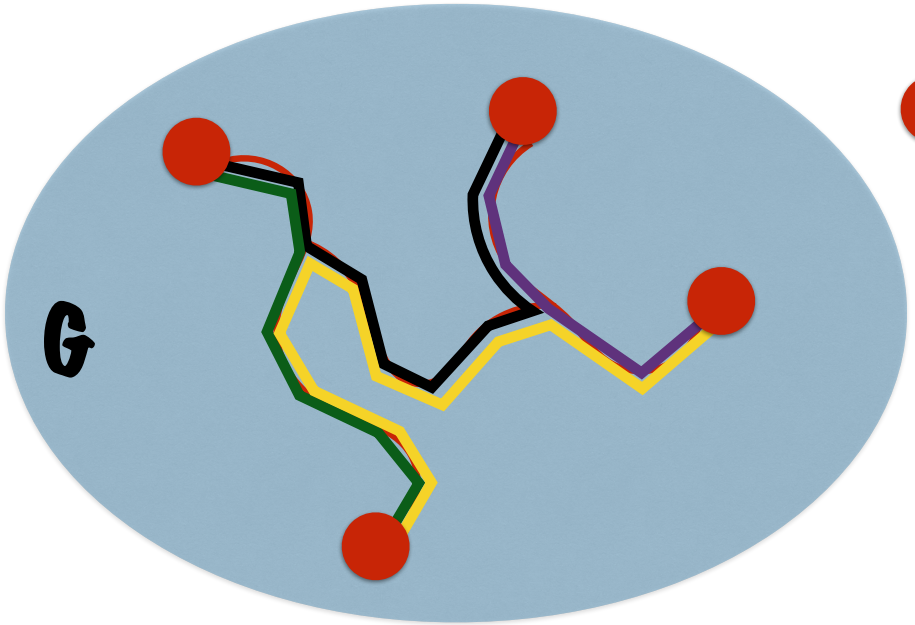
Theorem: it's a 2-approximation

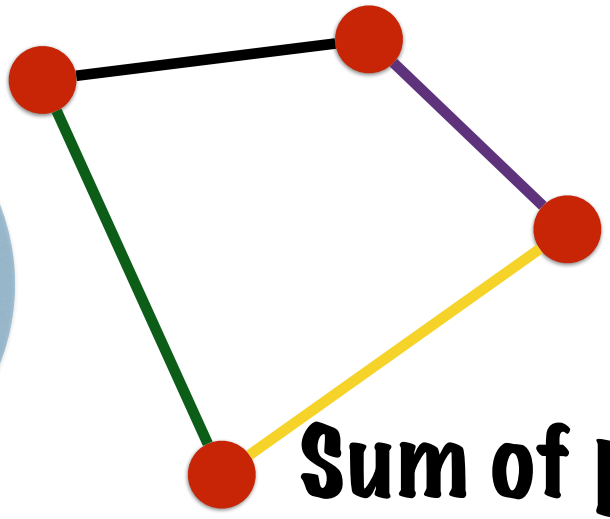
Go around the OPT tree
to define terminal-to-terminal paths



Theorem: it's a 2-approximation

Map terminal-to-terminal paths
to edges in complete graph of terminals

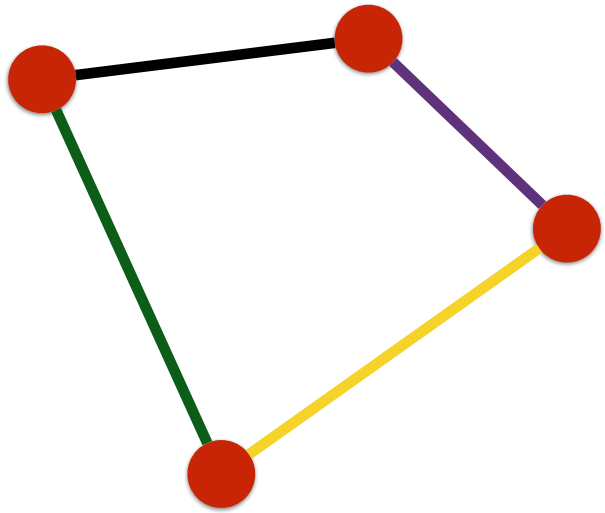


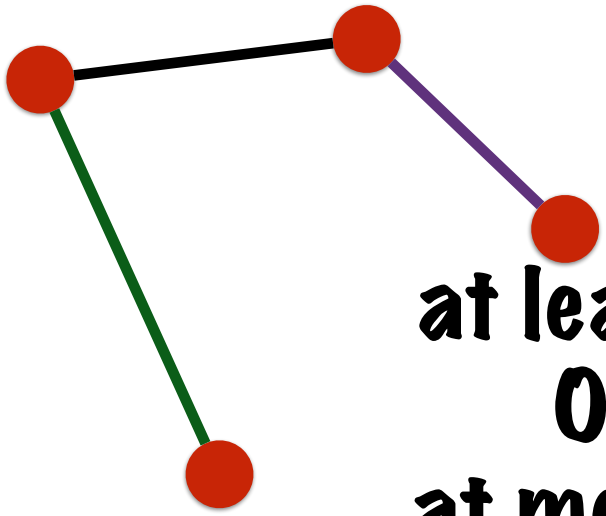


Sum of path lengths
= 2 OPT

Theorem: it's a 2-approximation

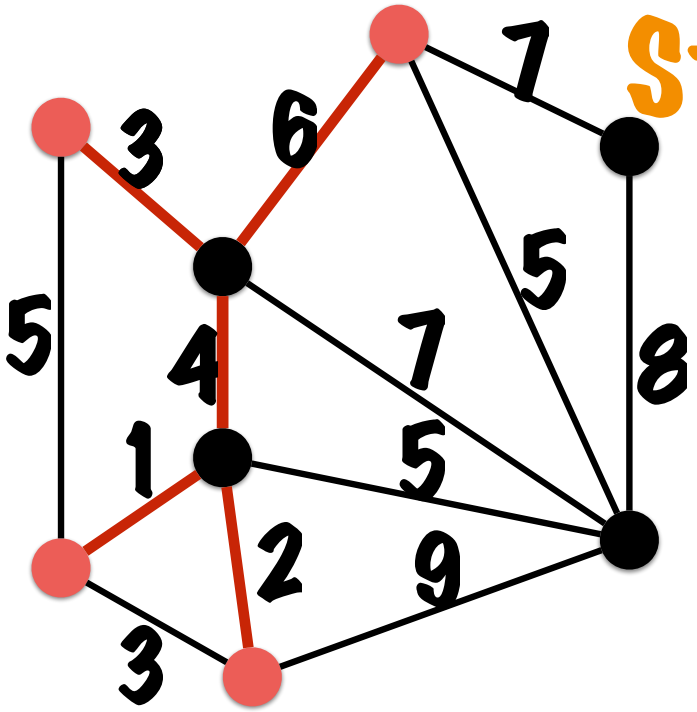
Remove one cycle edge





2OPT is
at least MST cost
Output cost is
at most MST cost
QED

Steiner tree



Jakob Steiner

Henry Pollak

H.N. Gilbert

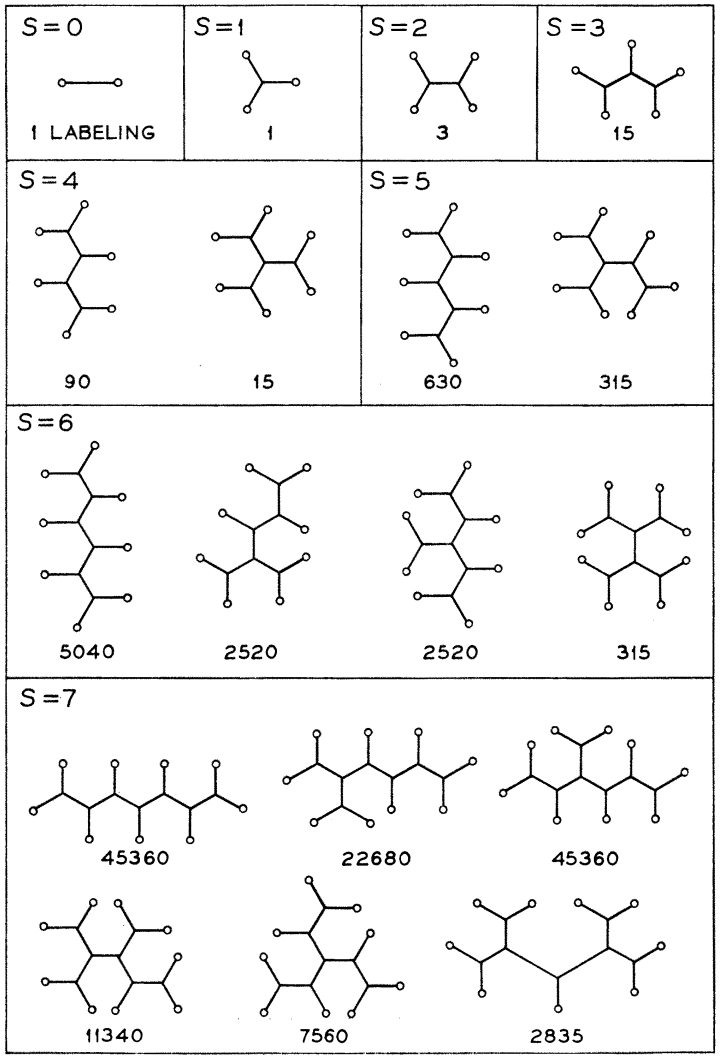
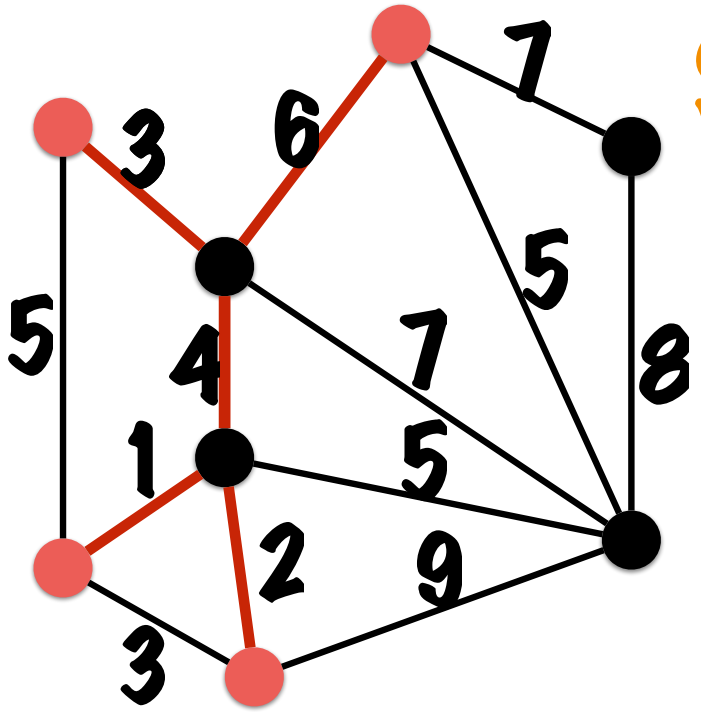


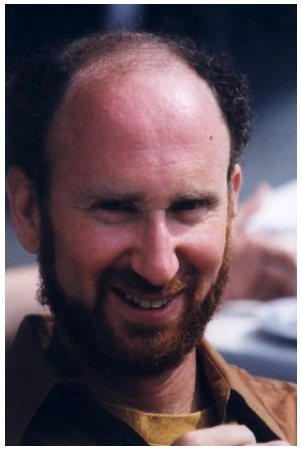
FIG. 3. *Full Steiner trees with s Steiner points*



Steiner tree

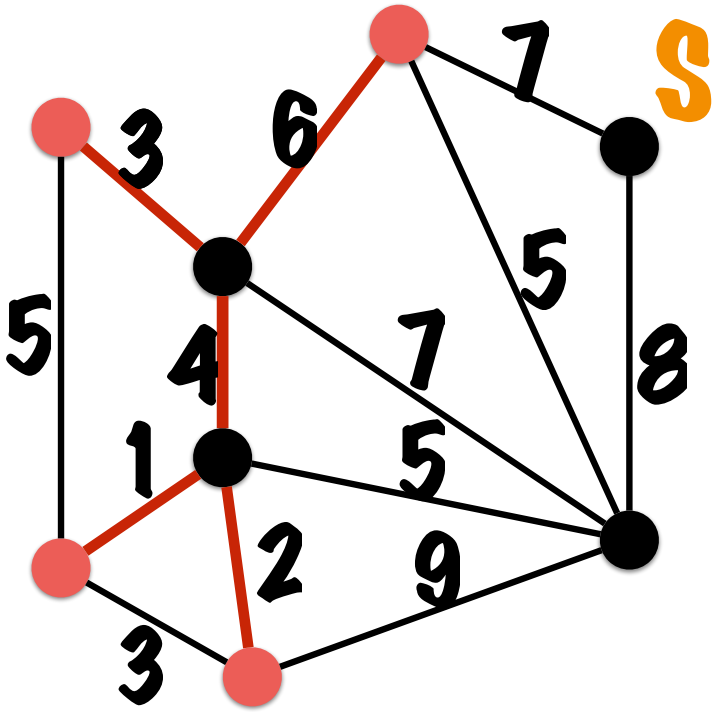
Jakob Steiner

Richard Karp



Marshall Bern

Paul Plassmann



Steiner tree 1.39

Jaroslaw Byrka

Fabrizio Grandoni

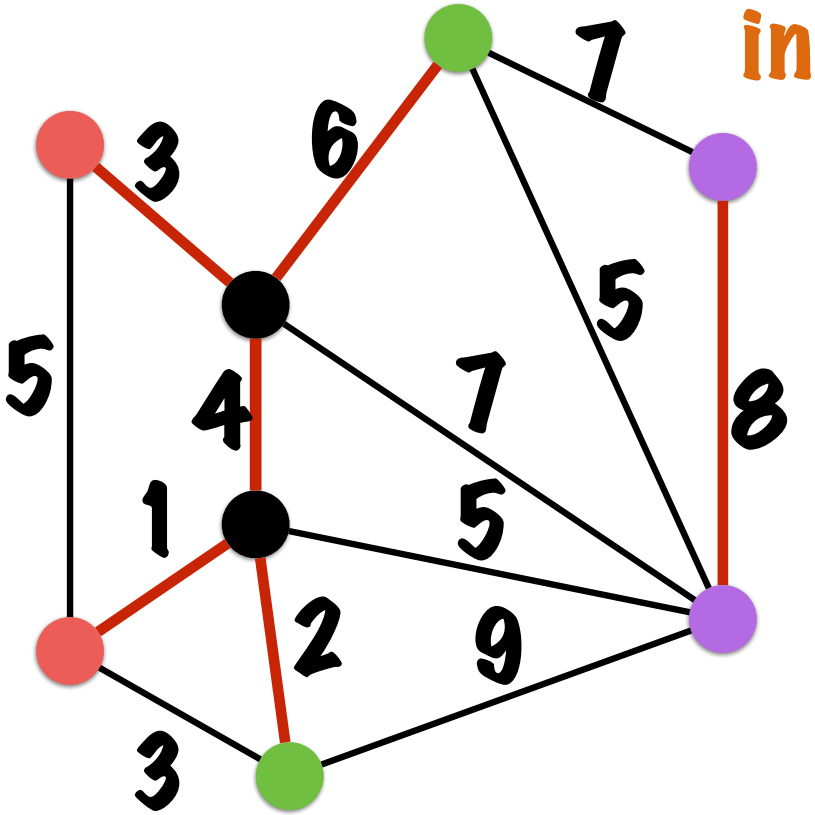
Laura Sanita

Thomas Rothvoss

Steiner forest

Steiner forest

Integer programming model



input: sets of terminals S1,S2,...

output: set of edges

min output cost

s.t.

for all i

for all u,v in Si

u and v are connected

Integer programming model

output: set of edges

$$x_e \in \{0, 1\}$$

min output cost

$$\min \sum_e c_e x_e$$

s.t.

for all i

$$\forall i$$

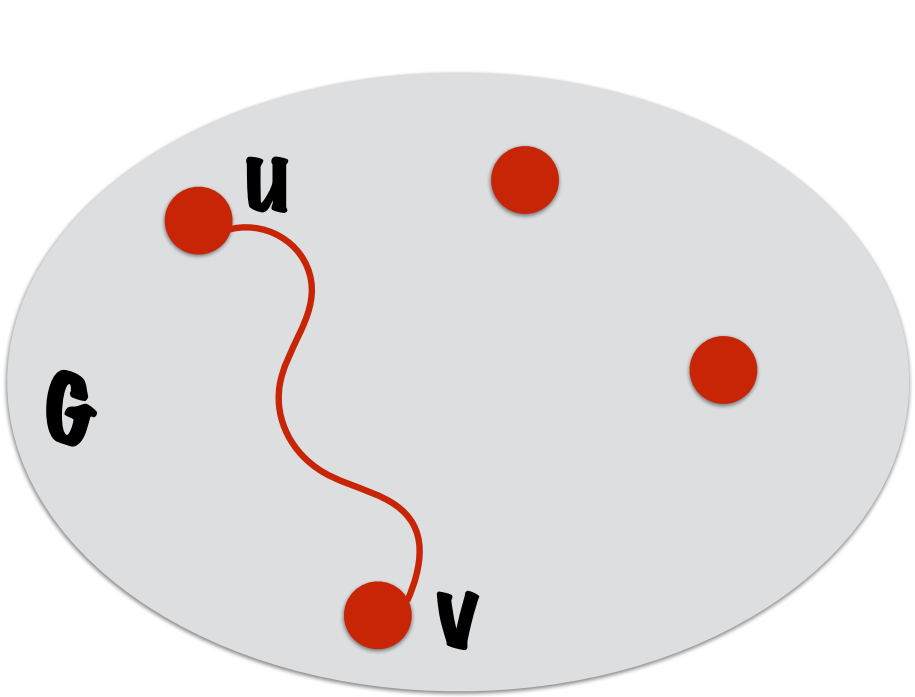
for all u,v in Si

$$\forall u, v \in S_i$$

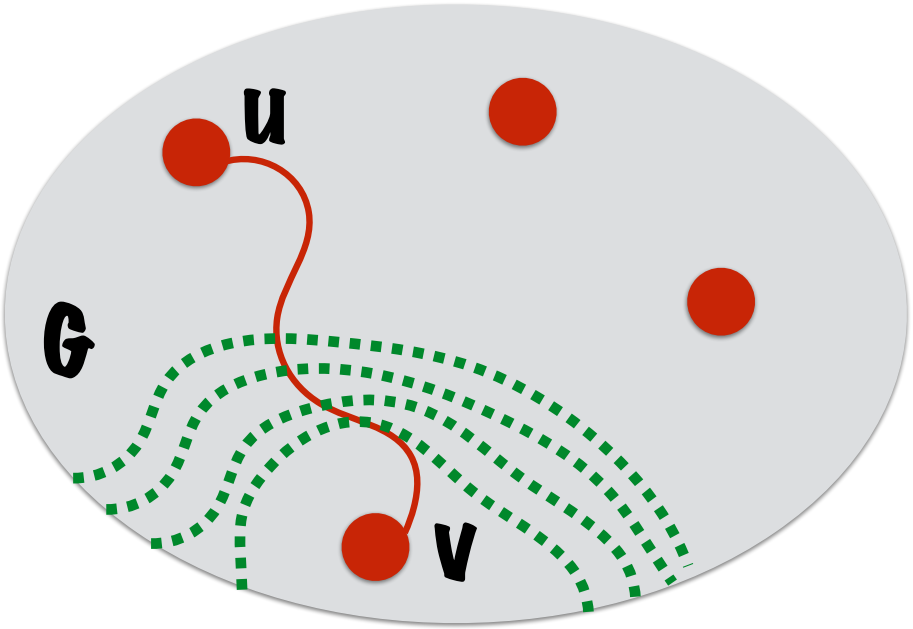
u and v are connected

?

Integer programming model

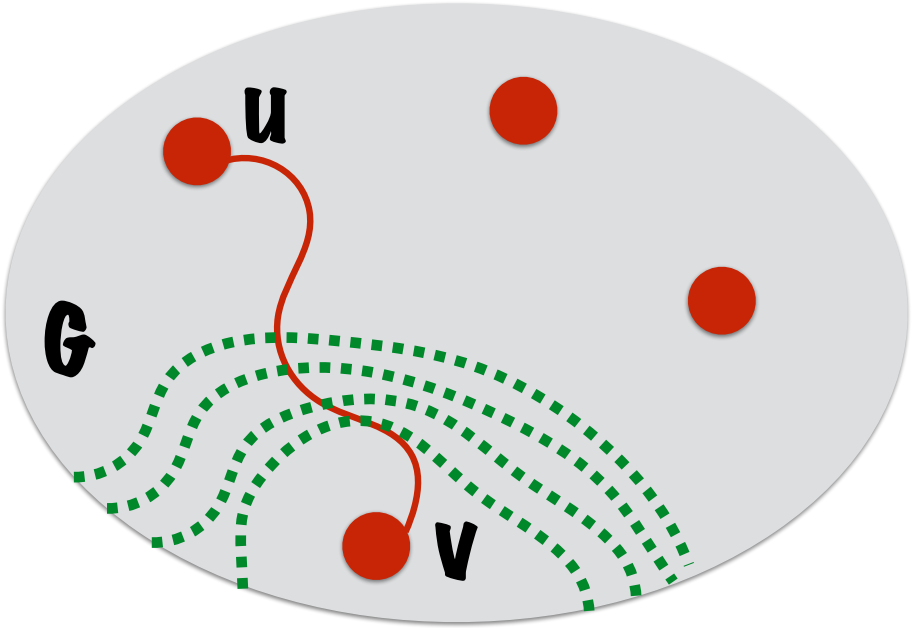


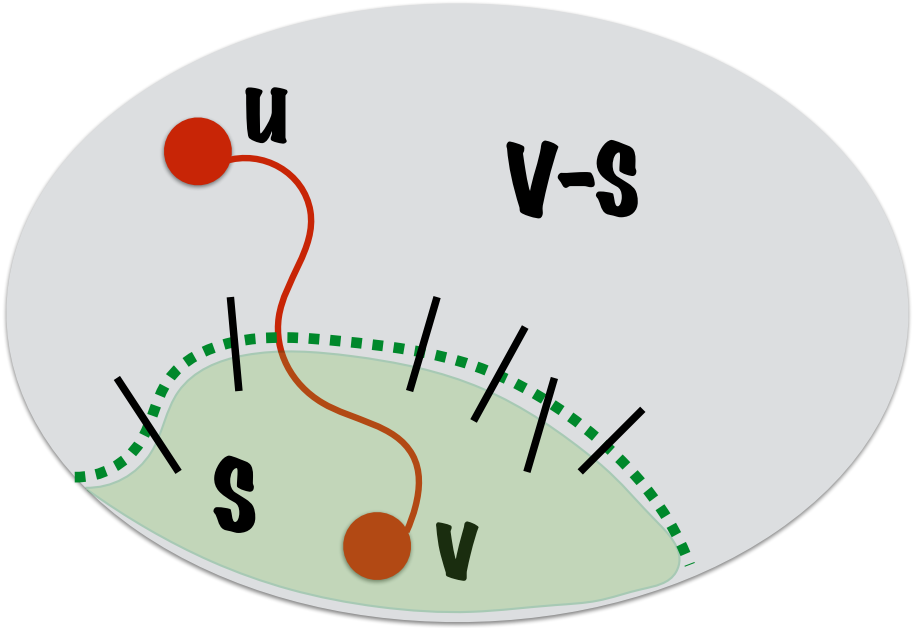
u and v are connected



every u-v cut is crossed

Integer programming model





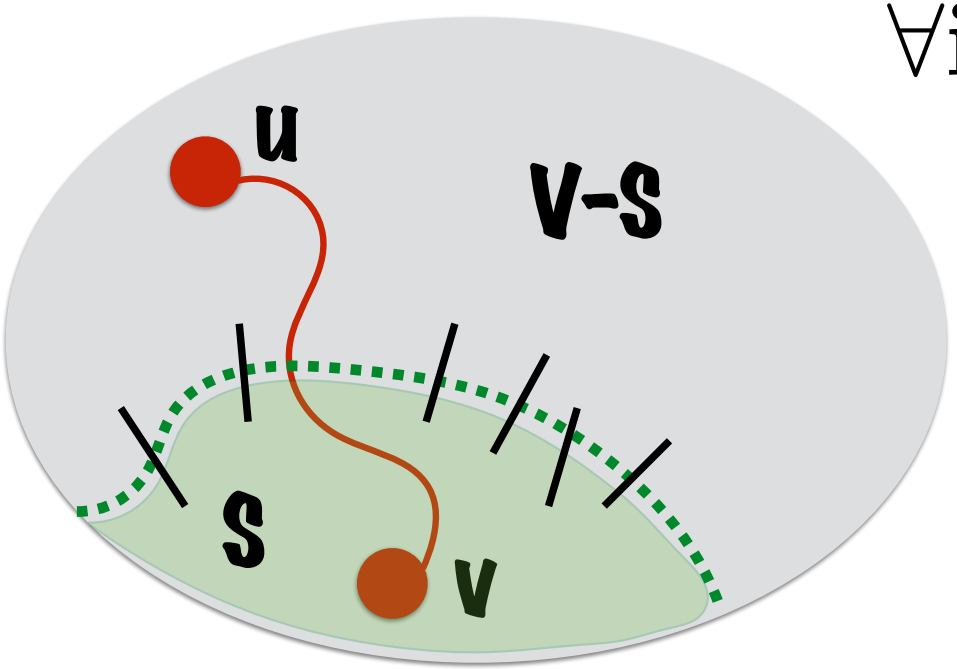
every u-v cut is crossed

$$\mathbf{S} \subseteq \mathbf{V} : |\mathbf{S} \cap \{\mathbf{u}, \mathbf{v}\}| = \mathbf{1}$$

$$\delta(\mathbf{S}) = \{\mathbf{e} \in \mathbf{S} \times \mathbf{V} \setminus \mathbf{S}\}$$

for all i
 for all u,v in Si
  u and v are connected

$$\mathbf{x_e} \in \{0, 1\}$$

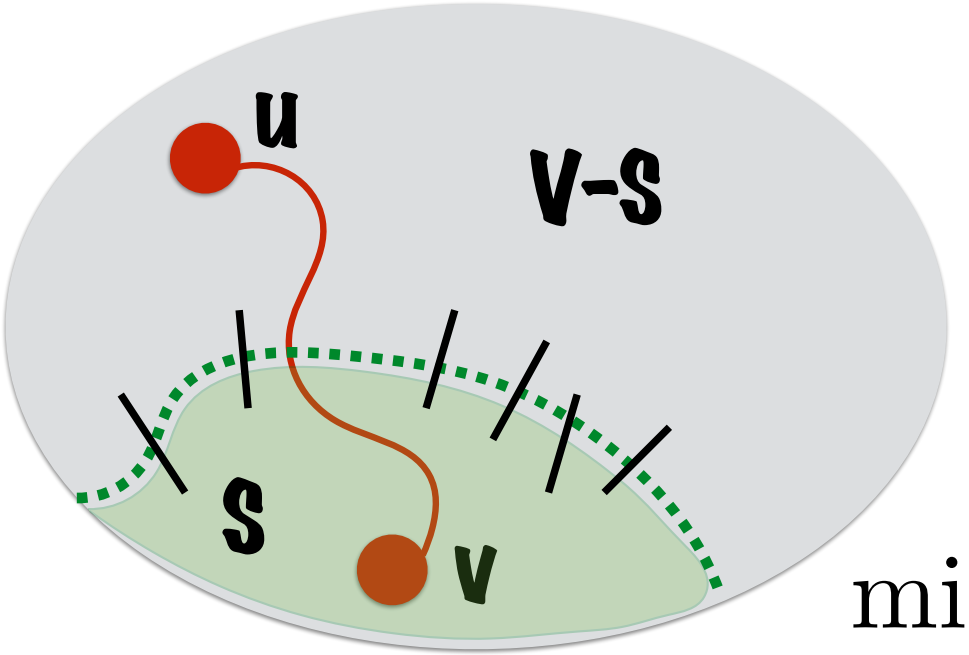


$$\forall \mathbf{i} \forall \mathbf{u}, \mathbf{v} \in \mathbf{S_i}$$

$$\forall \mathbf{S} \text{ s.t. } |\mathbf{S} \cap \{\mathbf{u}, \mathbf{v}\}| = 1$$

$$\sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq 1$$

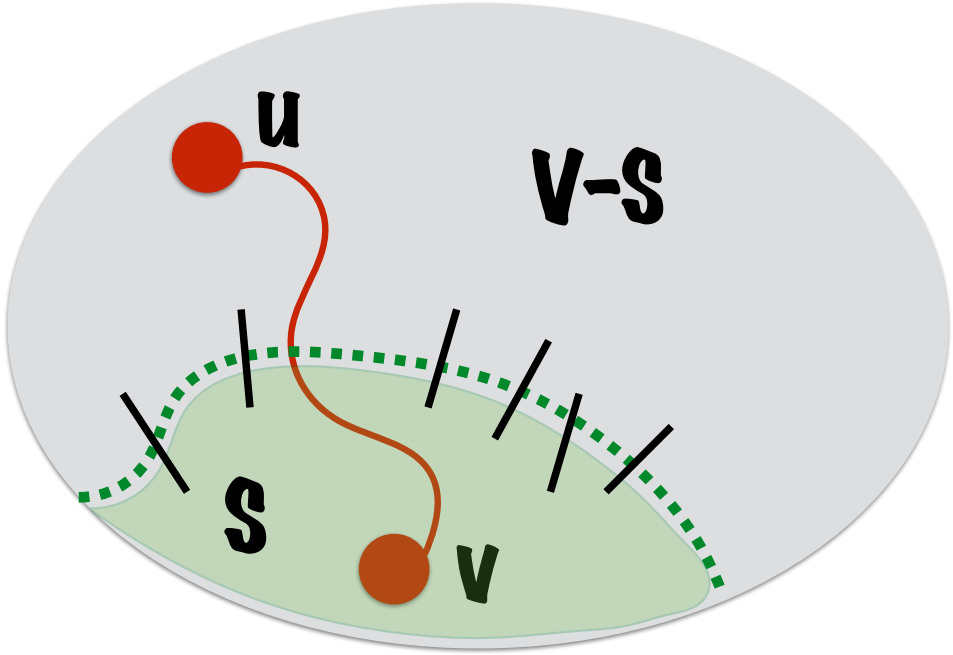
$$\mathcal{S} = \{\mathbf{S} : \exists \mathbf{i} \exists \mathbf{u}, \mathbf{v} \in \mathbf{S_i} : |\mathbf{S} \cap \{\mathbf{u}, \mathbf{v}\}| = \mathbf{1}\}$$



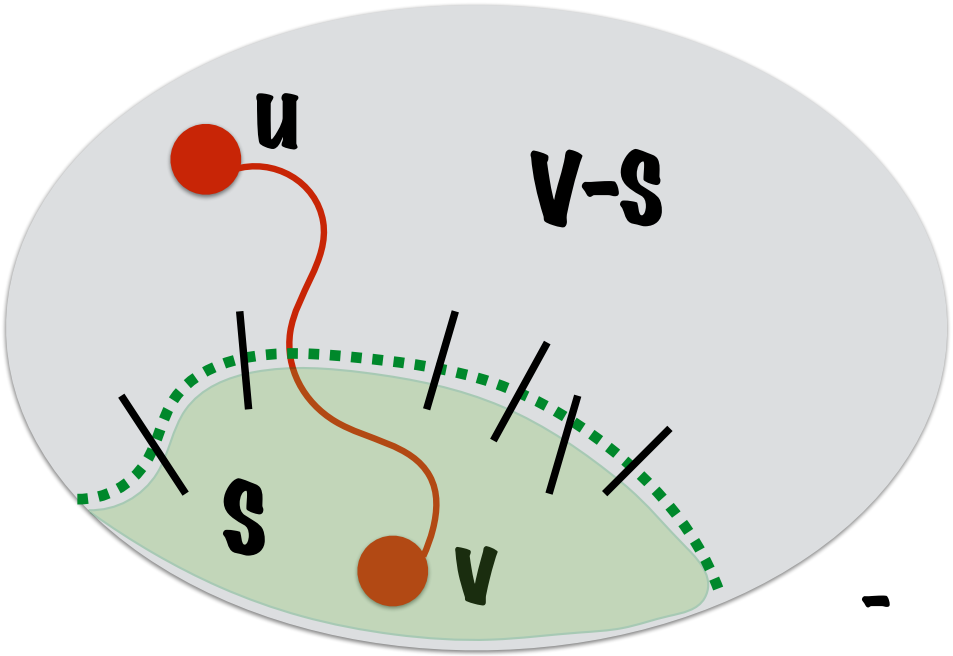
$$\min \sum_{\mathbf{e}} \mathbf{c_e x_e} :$$
$$\forall \mathbf{S} \in \mathcal{S} \sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1}$$
$$\mathbf{x_e} \in \{\mathbf{0}, \mathbf{1}\}$$

$\min \sum_{\mathbf{e}} \mathbf{c_e x_e}$:
$\forall \mathbf{S} \in \mathcal{S} \sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1}$
$\mathbf{x_e} \in \{\mathbf{0}, \mathbf{1}\}$

$\min \sum_{\mathbf{e}} \mathbf{c_e x_e}$:
$\forall \mathbf{S} \in \mathcal{S} \sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1}$
$\mathbf{x_e} \geq \mathbf{0}$



$$\min \sum_e c_e x_e :$$
$$\forall S \in \mathcal{S} \sum_{e \in \delta(S)} x_e \geq 1$$
$$x_e \geq 0$$



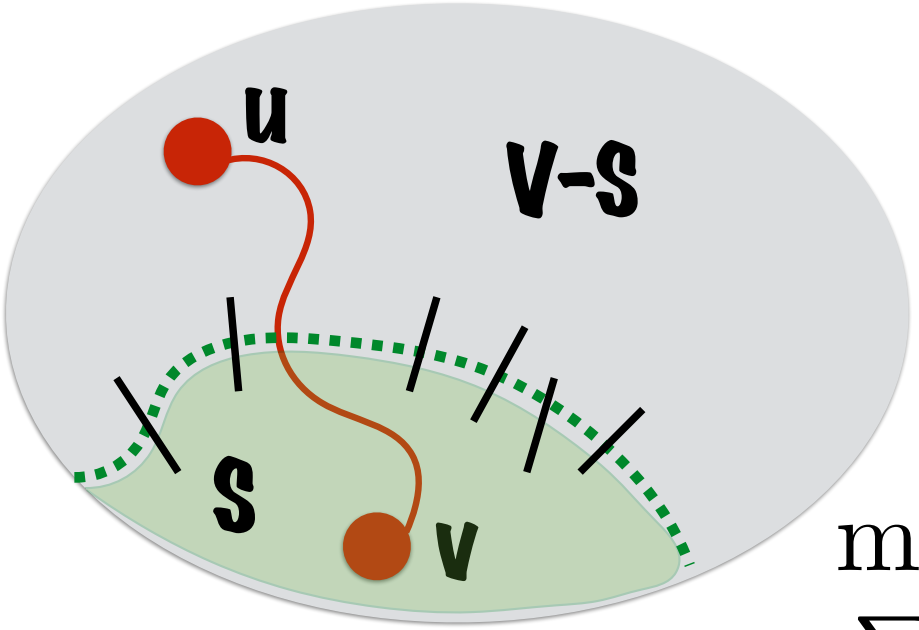
Exponential
number of constraints
but
- still solvable
- does not need to be solved

Steiner forest

Steiner forest

Linear programming relaxation

$$\mathcal{S} = \{\mathbf{S} : \exists \mathbf{i} \exists \mathbf{u}, \mathbf{v} \in \mathbf{S_i} : |\mathbf{S} \cap \{\mathbf{u}, \mathbf{v}\}| = \mathbf{1}\}$$



$$\min \sum_{\mathbf{e}} \mathbf{c_e x_e} :$$
$$\sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1} \quad \forall \mathbf{S} \in \mathcal{S}$$
$$\mathbf{x_e} \geq \mathbf{0} \quad \forall \mathbf{e} \in \mathbf{E}$$

Taking the dual

$$\min \sum_{\mathbf{e}} \mathbf{c_e x_e} :$$

$$\sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1} \quad \forall \mathbf{S} \in \mathcal{S} \quad [\mathbf{y_S}]$$

$$\mathbf{x_e} \geq \mathbf{0} \quad \forall \mathbf{e} \in \mathbf{E}$$

$$\max \sum_{\mathbf{S}} \mathbf{y_S} :$$

$$\sum_{\mathbf{S}:\mathbf{e} \in \delta(\mathbf{S})} \mathbf{y_S} \leq \mathbf{c_e} \quad \forall \mathbf{e} \in \mathbf{E} \quad [\mathbf{x_e}]$$

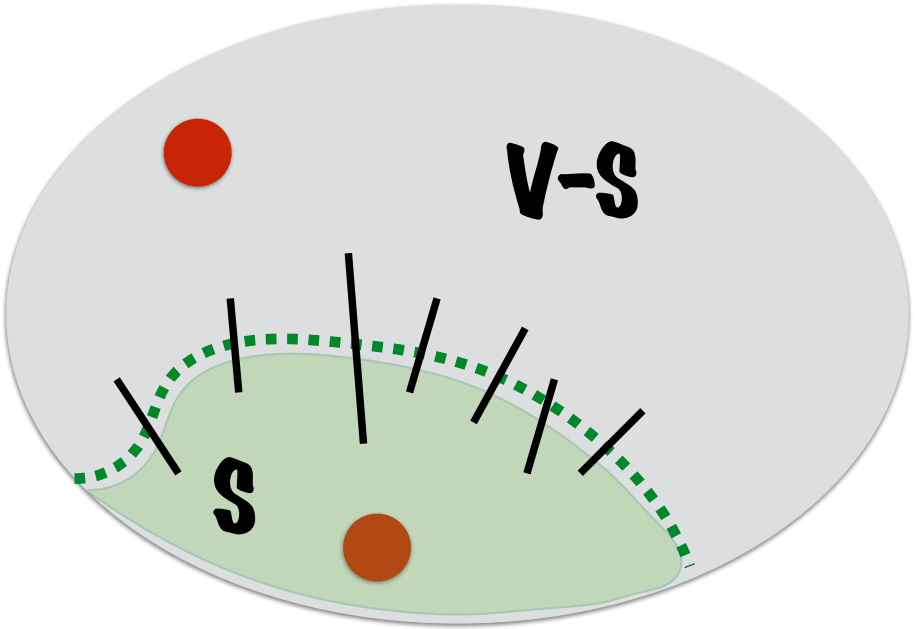
$$\mathbf{y_S} \geq \mathbf{0} \quad \forall \mathbf{S} \in \mathcal{S}$$

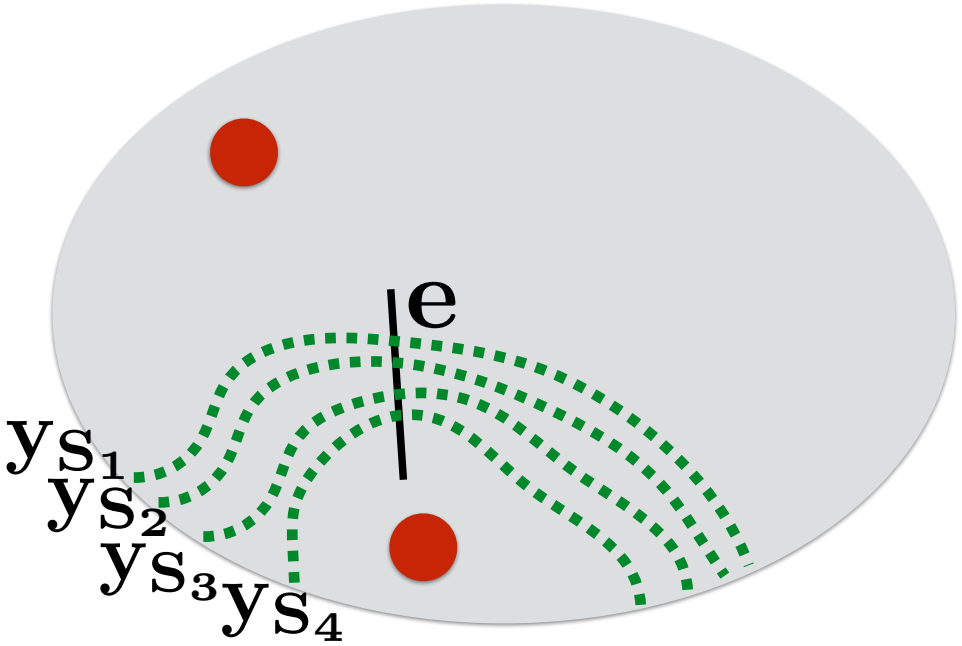
Interpreting the dual

$$\max \sum_S y_S :$$

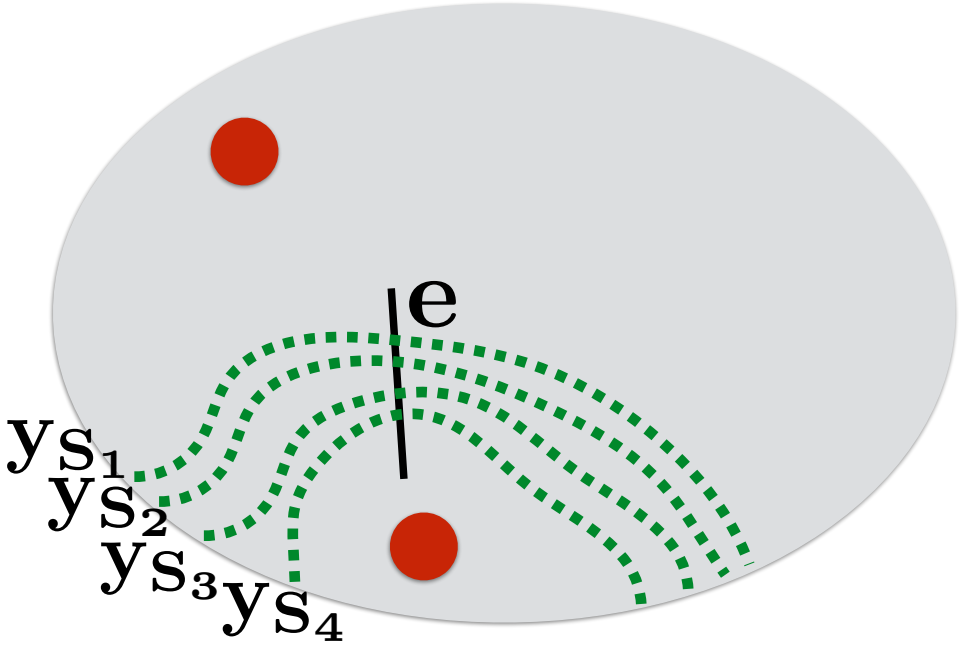
$$\sum_{S:e\in\delta(S)} y_S \le c_e \quad \forall e \in E \quad [x_e]$$

$$y_S \ge 0 \quad \forall S \in \mathcal{S}$$





$$\max \sum_{S} y_S :$$
$$\sum_{S:e\in\delta(S)} y_S \le c_e \quad \forall e \in E \quad [x_e]$$
$$y_S \ge 0 \quad \forall S \in \mathcal{S}$$



Sooo many cuts containing e!!
Q: How can we hope for feasibility?
A: most variables will be 0

Steiner forest

Steiner forest

primal-dual pair

$$\min \sum_{\mathbf{e}} \mathbf{c_e x_e} :$$

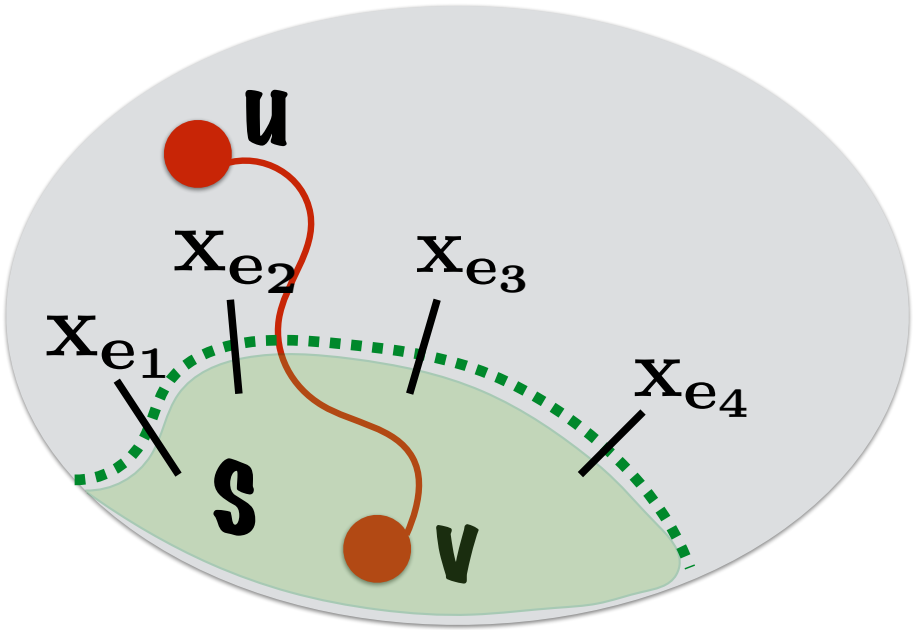
$$\sum_{\mathbf{e} \in \delta(\mathbf{S})} \mathbf{x_e} \geq \mathbf{1} \quad \forall \mathbf{S} \in \mathcal{S} \quad [\mathbf{y_S}]$$

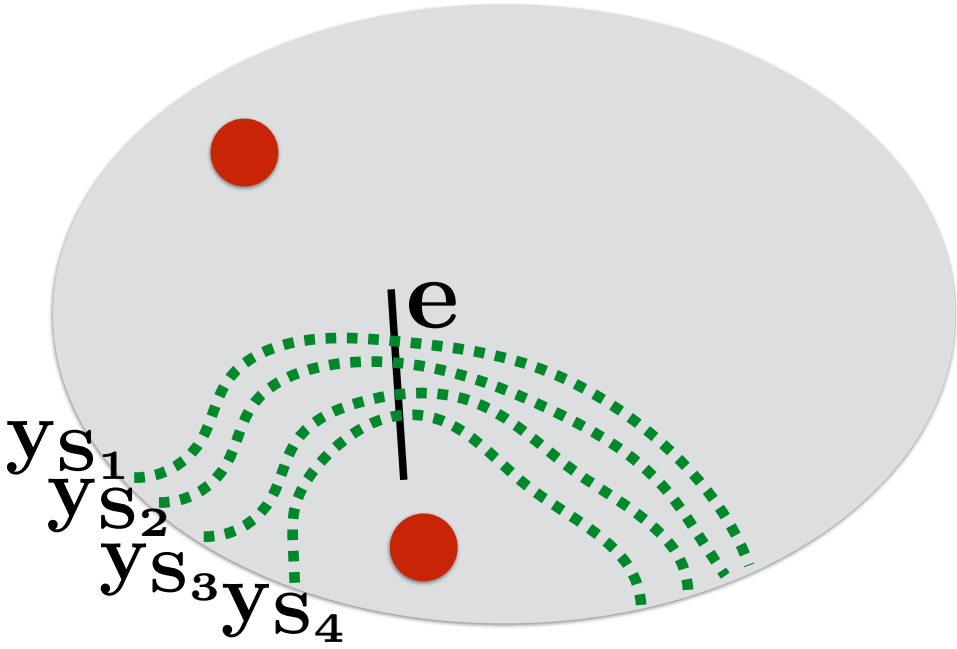
$$\mathbf{x_e} \geq \mathbf{0} \quad \forall \mathbf{e} \in \mathbf{E}$$

$$\max \sum_{\mathbf{S}} \mathbf{y_S} :$$

$$\sum_{\mathbf{S}:\mathbf{e} \in \delta(\mathbf{S})} \mathbf{y_S} \leq \mathbf{c_e} \quad \forall \mathbf{e} \in \mathbf{E} \quad [\mathbf{x_e}]$$

$$\mathbf{y_S} \geq \mathbf{0} \quad \forall \mathbf{S} \in \mathcal{S}$$





$$\sum_{e \in \delta(S)} x_e \geq 1$$

$$\sum_{S : e \in \delta(S)} y_S \leq c_e$$

$\min \sum_e c_e x_e$:
$\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S}$
$x_e \geq 0 \quad \forall e \in E$

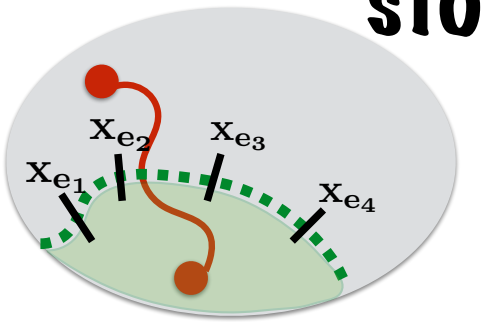
$\max \sum_S y_S$:
$\sum_{S : e \in \delta(S)} y_S \leq c_e \quad \forall e \in E$
$y_S \geq 0 \quad \forall S \in \mathcal{S}$

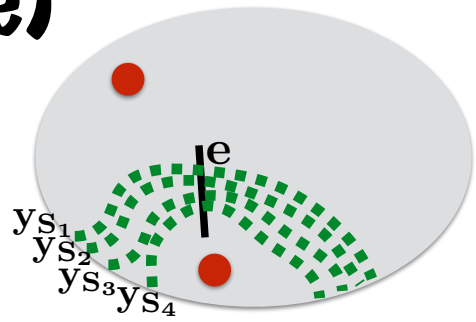
Initialization:

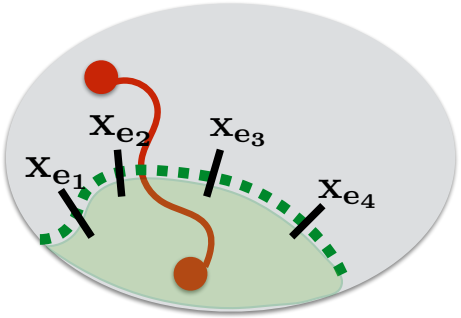
$$x \leftarrow 0, y \leftarrow 0$$

Iteration: while x not satisfiable
raise y as much as possible
stopped by tight constraint (e)

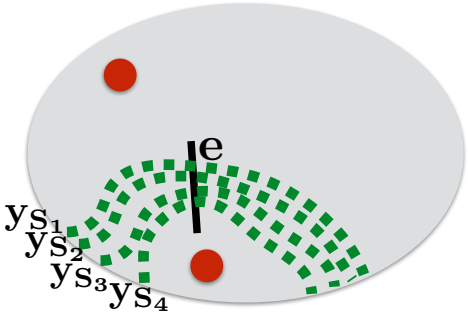
$$x_e \leftarrow 1$$



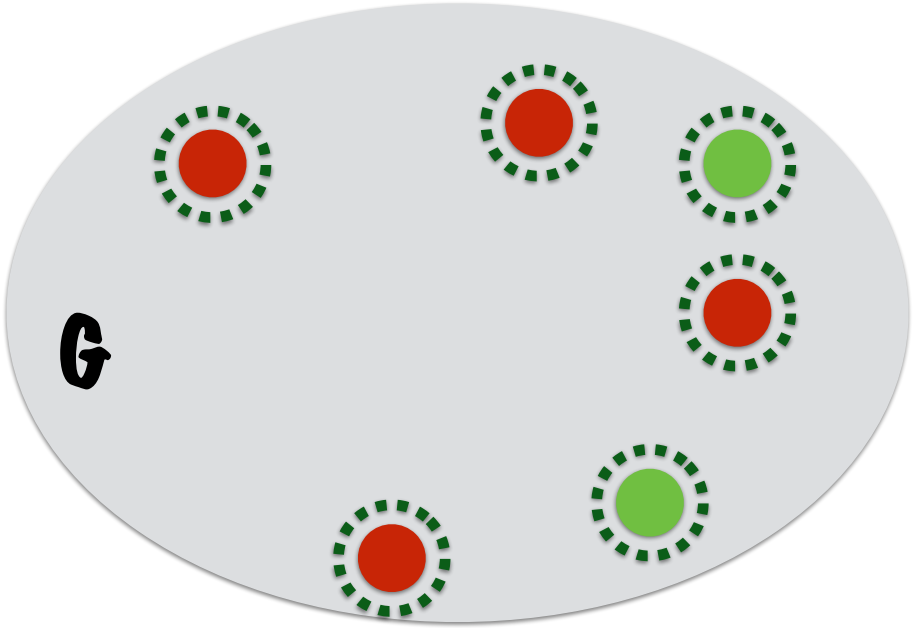




Q: How do we raise y?



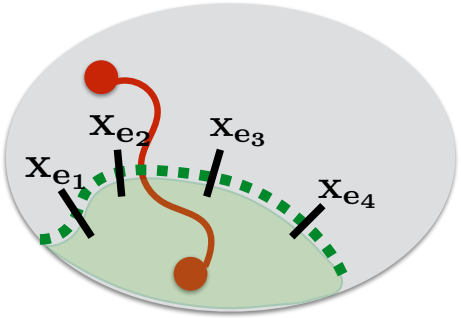
Initially...

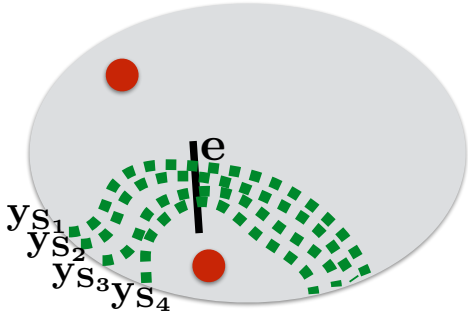


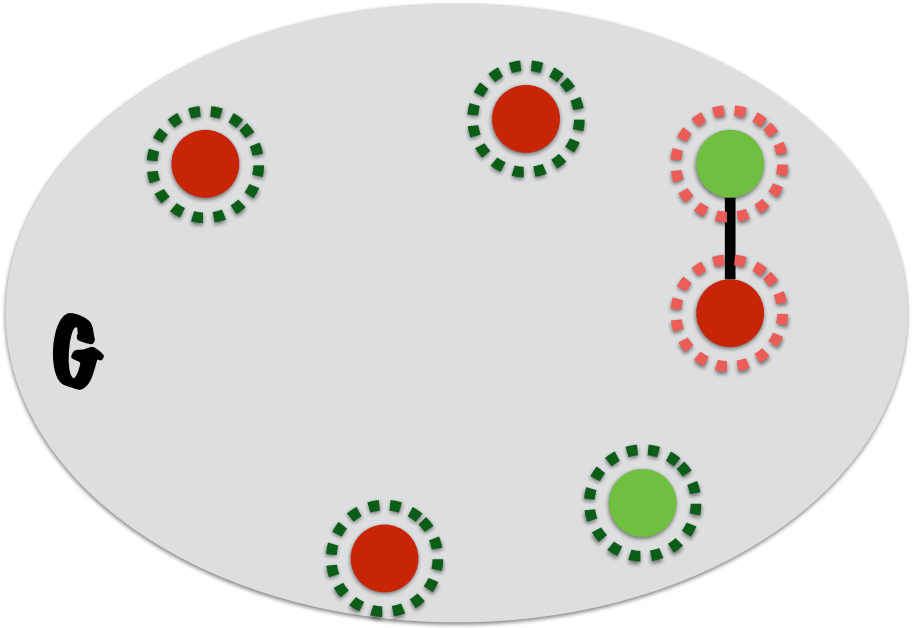
Raise all singleton terminals simultaneously

$$y_S \leftarrow y_S + \epsilon \quad \forall S = \{u\}$$

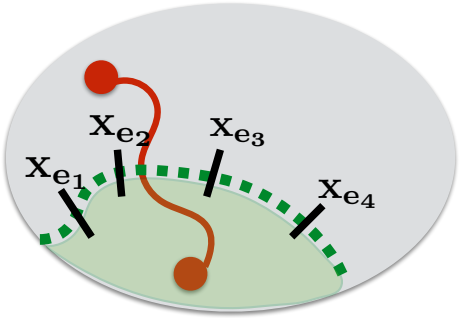
Q: How do we raise y?



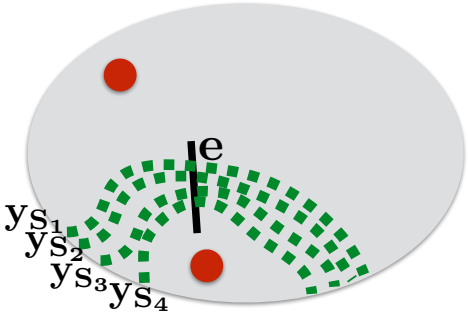




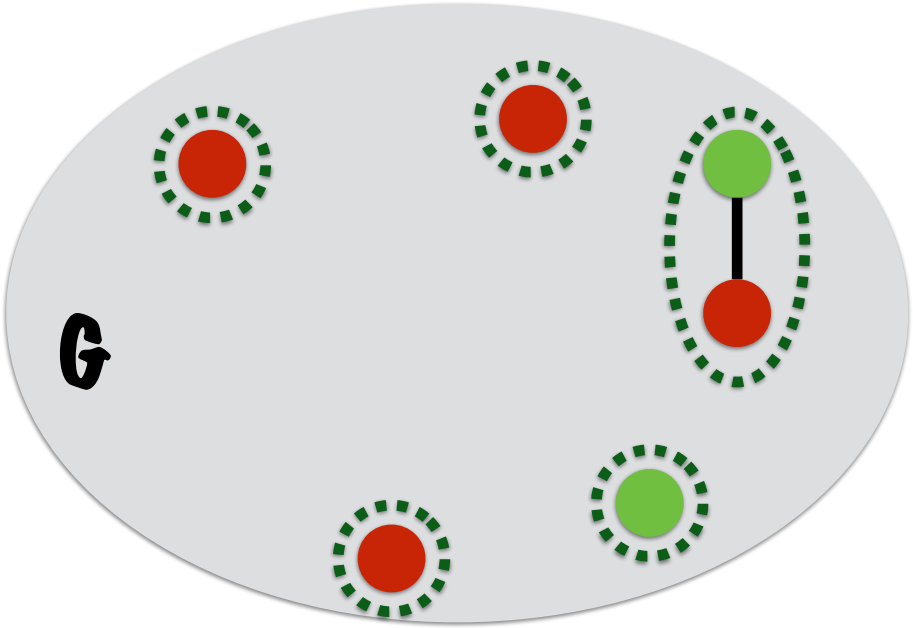
Tight constraint...



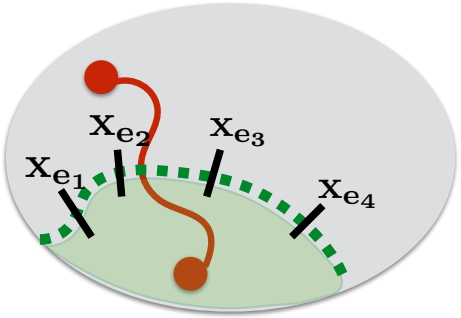
Q: How do we raise y?

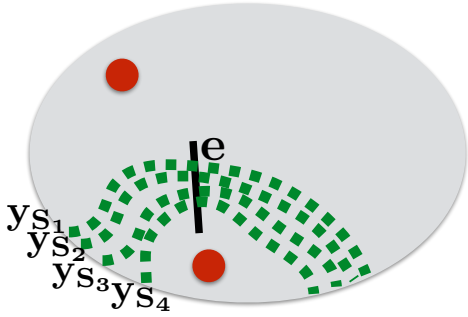


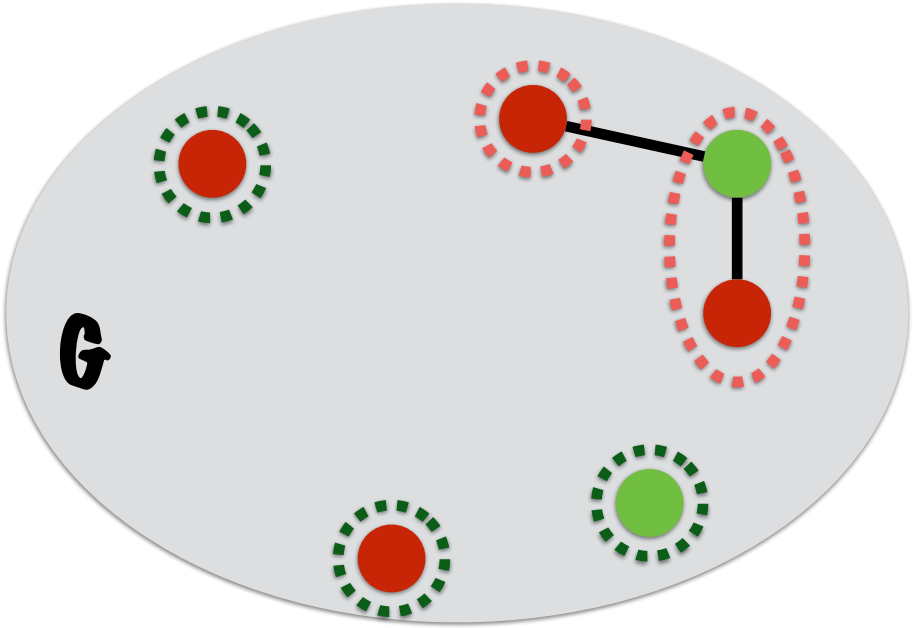
Then...



Q: How do we raise y?

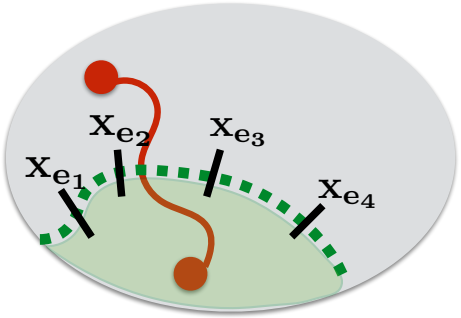


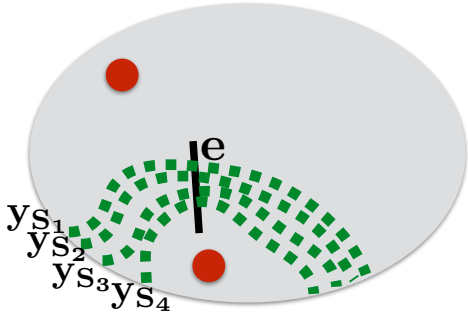


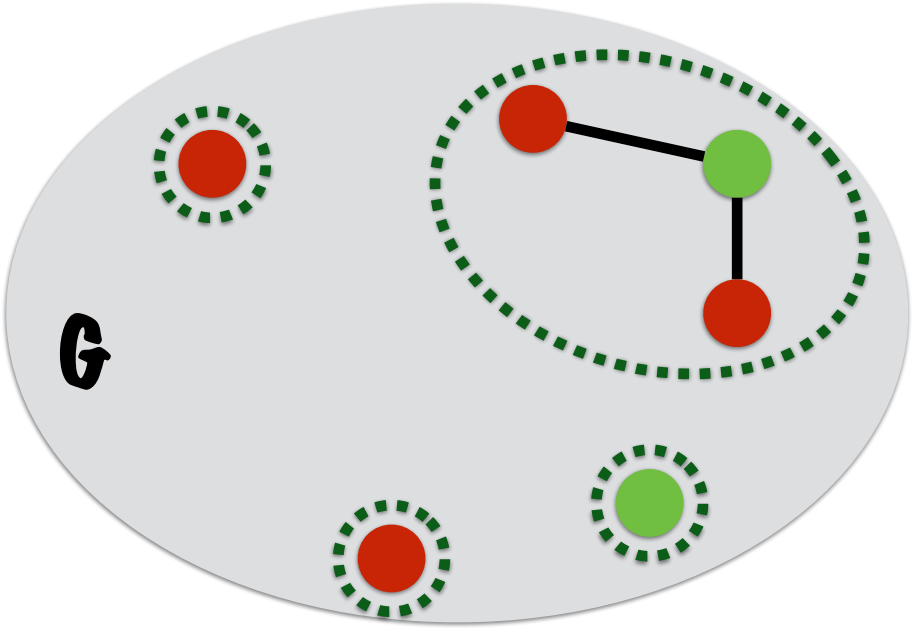


Tight constraint...

Q: How do we raise y?

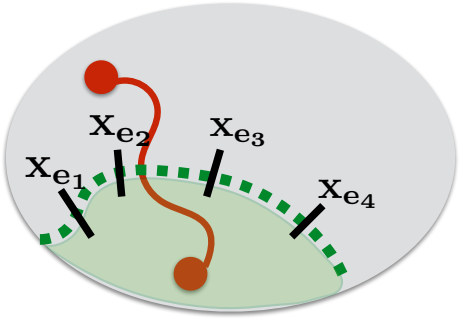


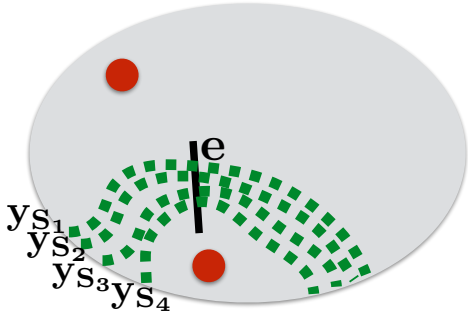


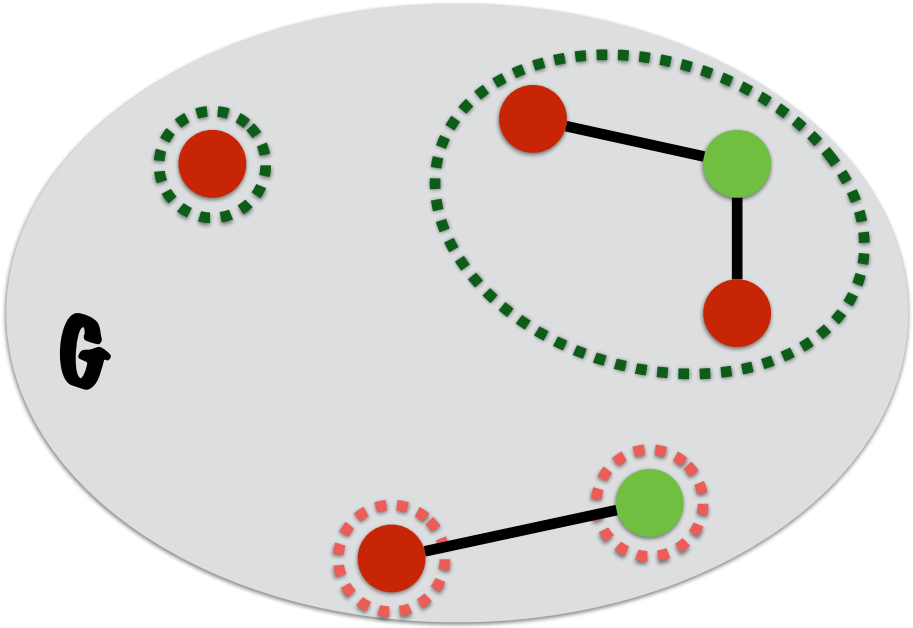


Then...

Q: How do we raise y?

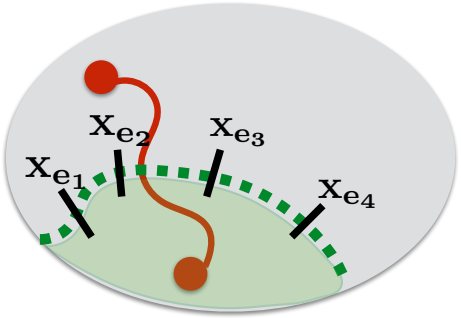


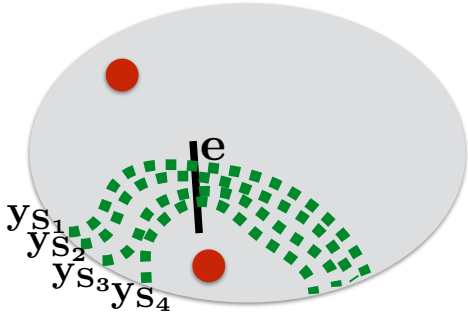


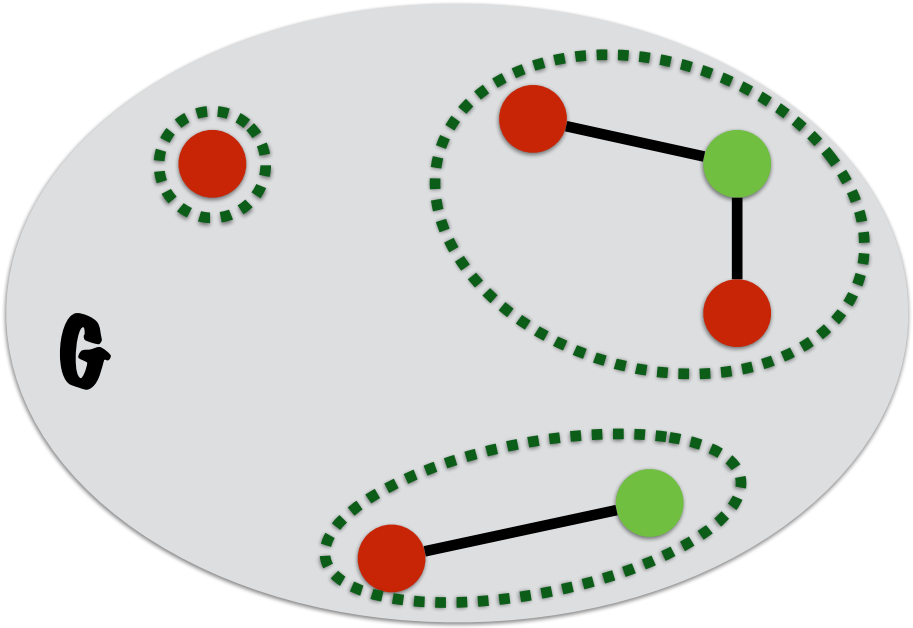


Tight constraint...

Q: How do we raise y?

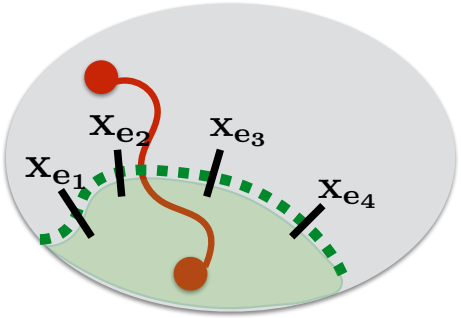


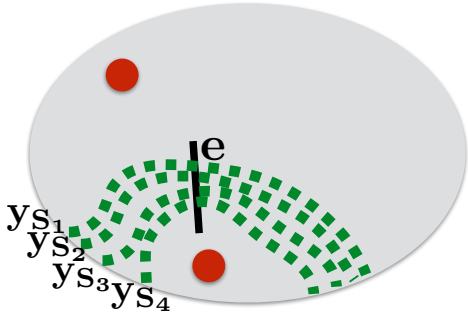


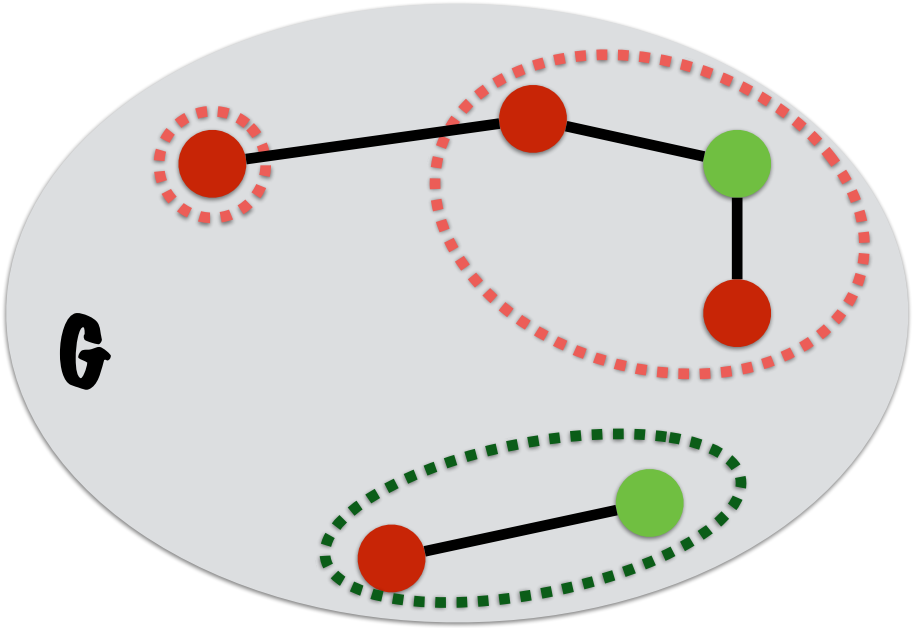


Then...

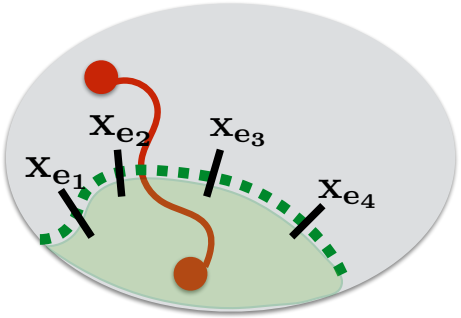
Q: How do we raise y?



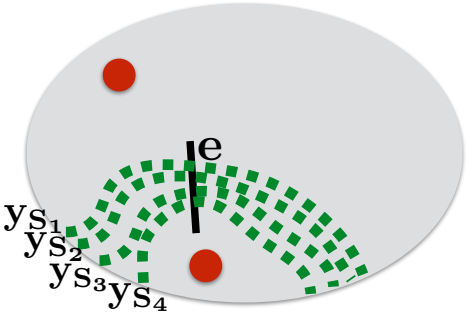


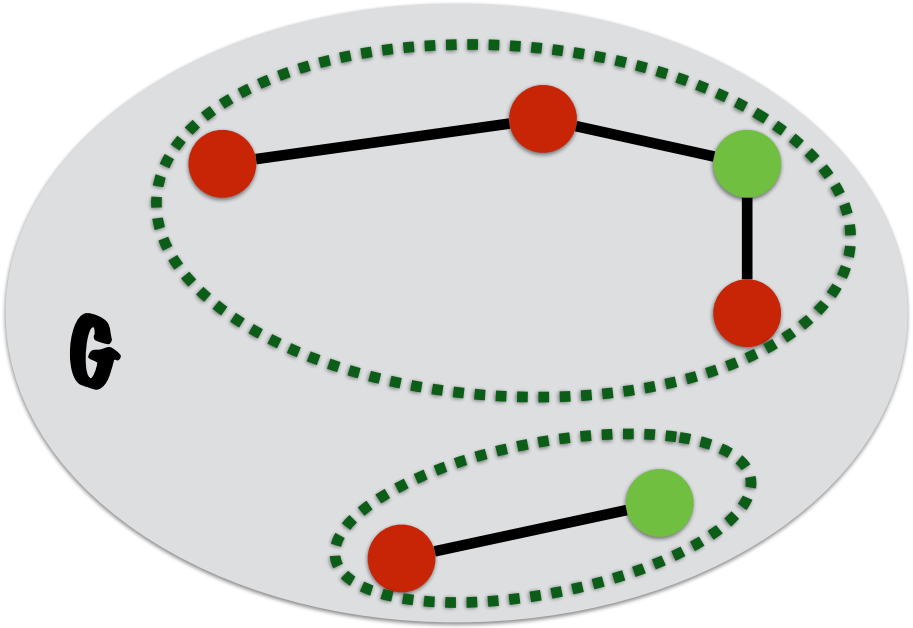


Tight constraint...



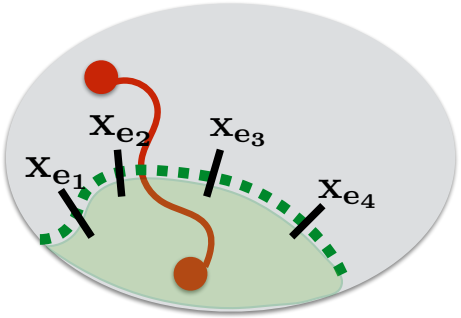
Q: How do we raise y?

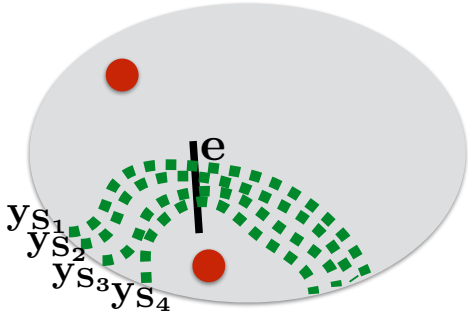


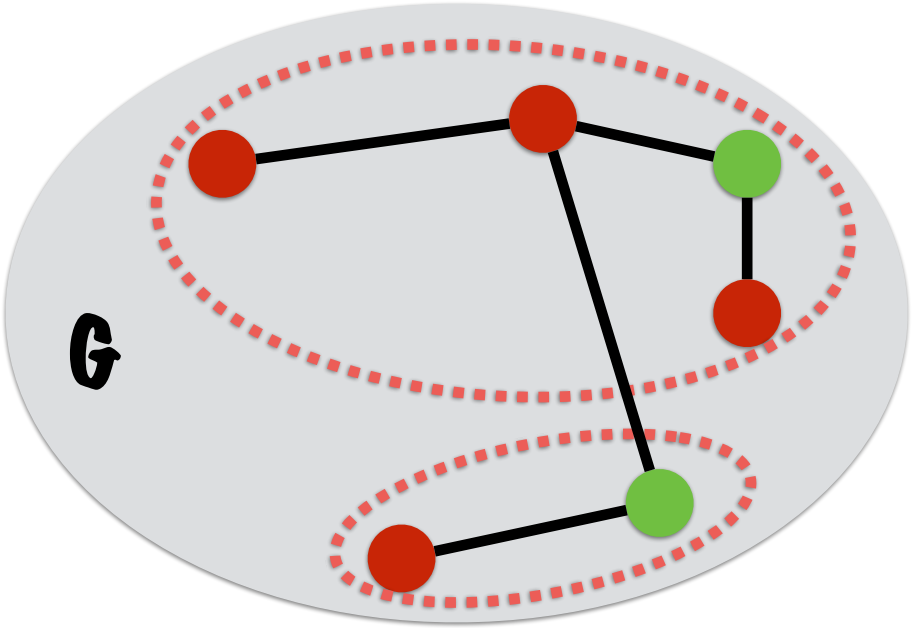


Then...

Q: How do we raise y?

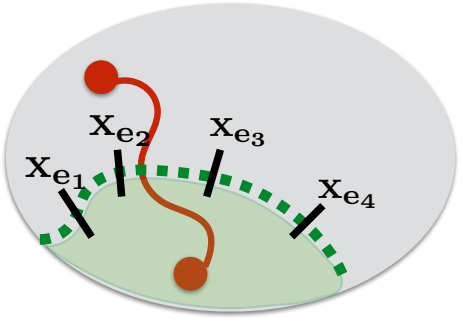


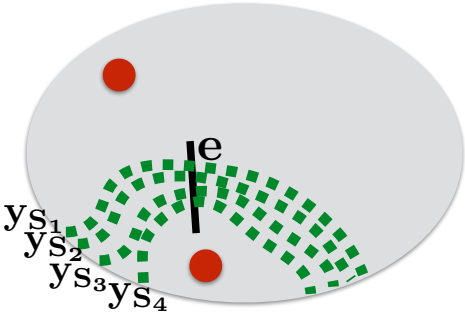


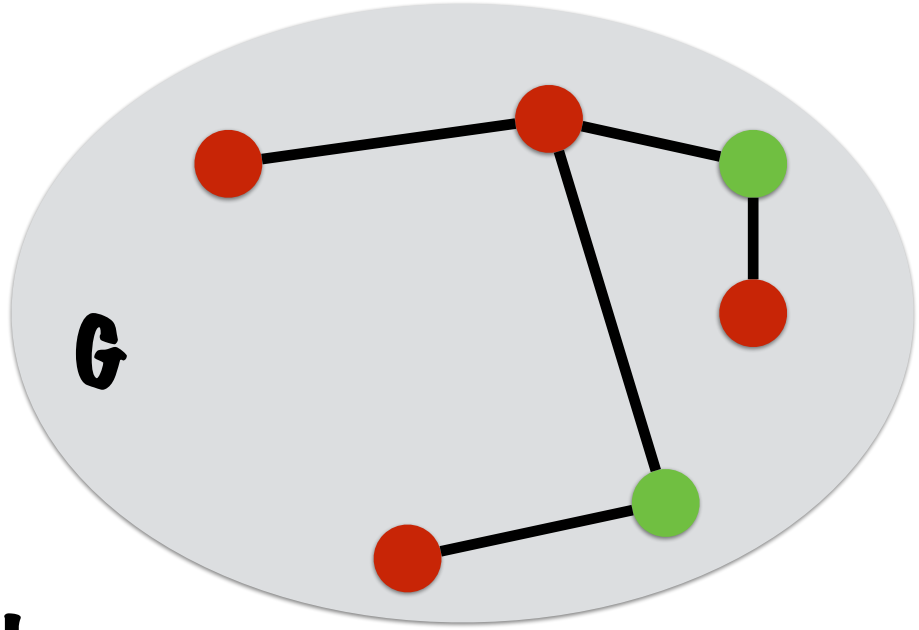


Tight constraint...

Q: How do we raise y?

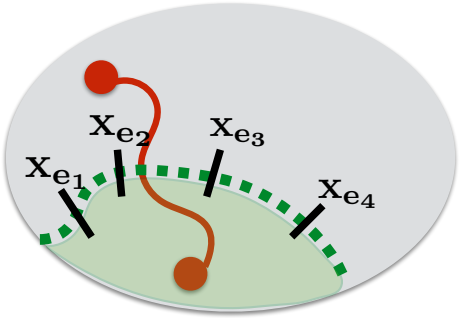


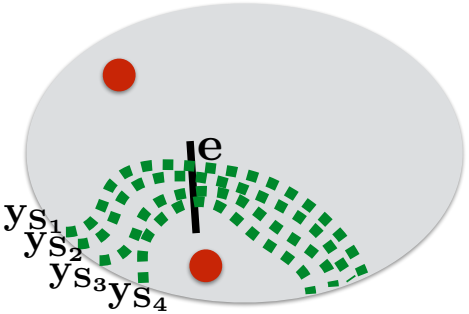


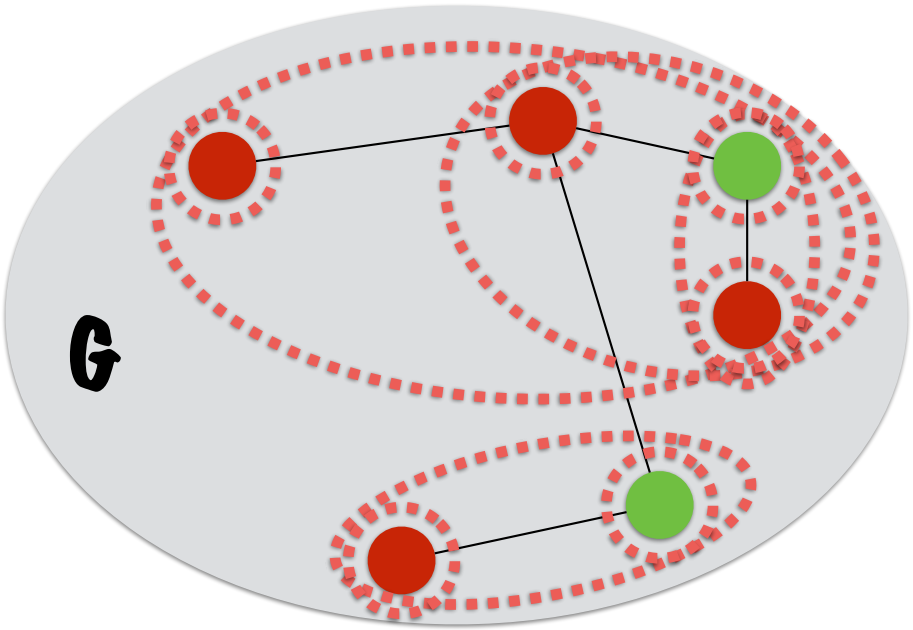


x is feasible

Q: How did we raise y?







$\min \sum_e c_e x_e$:

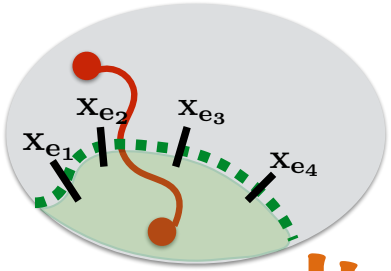
$\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S}$

$x_e \geq 0 \quad \forall e \in E$

$\max \sum_S y_S$:

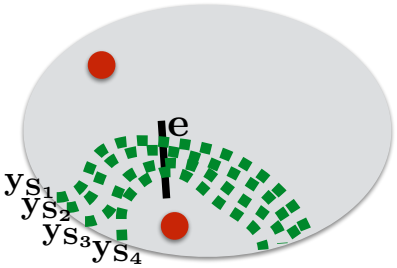
$\sum_{S: e \in \delta(S)} y_S \leq c_e \quad \forall e \in E$

$y_S \geq 0 \quad \forall S \in \mathcal{S}$



Initialization:

$$x \leftarrow 0, y \leftarrow 0$$



Iteration: while x not satisfiable

in parallel, raise every unfrozen y_S with minimal S

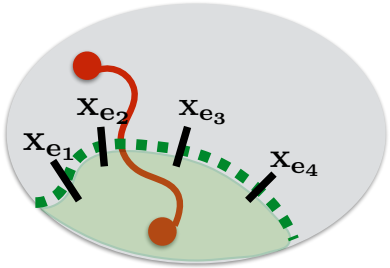
stopped by tight constraint (e)

$$x_e \leftarrow 1$$

freeze y_S in tight constraints

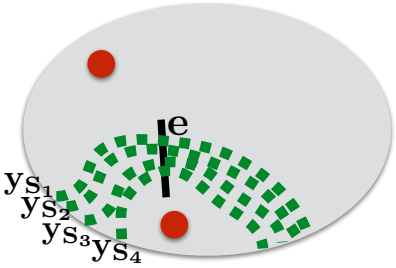
Steiner forest

Steiner forest



Initialization:

$$x \leftarrow 0, y \leftarrow 0$$



Iteration: while x not satisfiable
in parallel, raise every unfrozen y_S with minimal S
stopped by tight constraint (e)

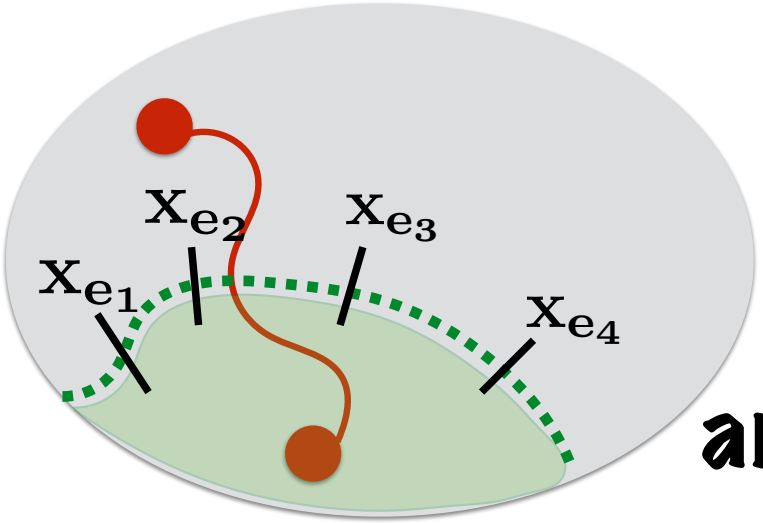
$$x_e \leftarrow 1$$

freeze y_S in tight constraints

Do we ever get stuck?

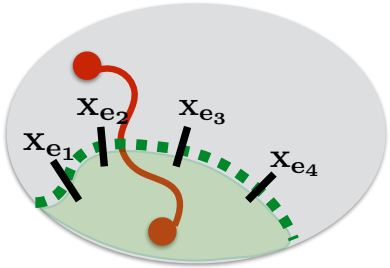
Do we ever get stuck?
Suppose we do

cut: x not feasible?



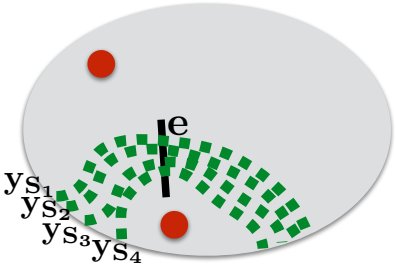
If we cannot raise y(S)
it's because
some e in the cut is tight
and then we would have put x(e)
in solution

QED



Initialization:

$$x \leftarrow 0, y \leftarrow 0$$



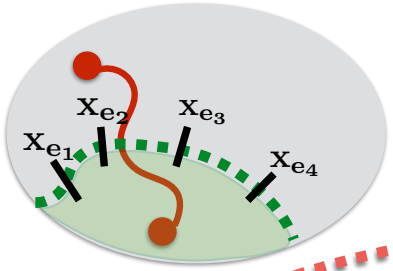
Iteration: while x not satisfiable
in parallel, raise every unfrozen y_S with minimal S
stopped by tight constraint (e)

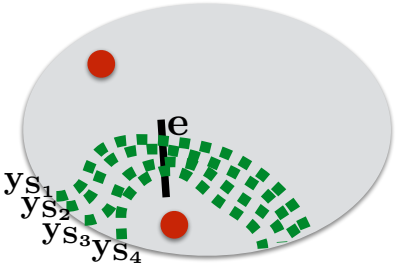
$$x_e \leftarrow 1$$

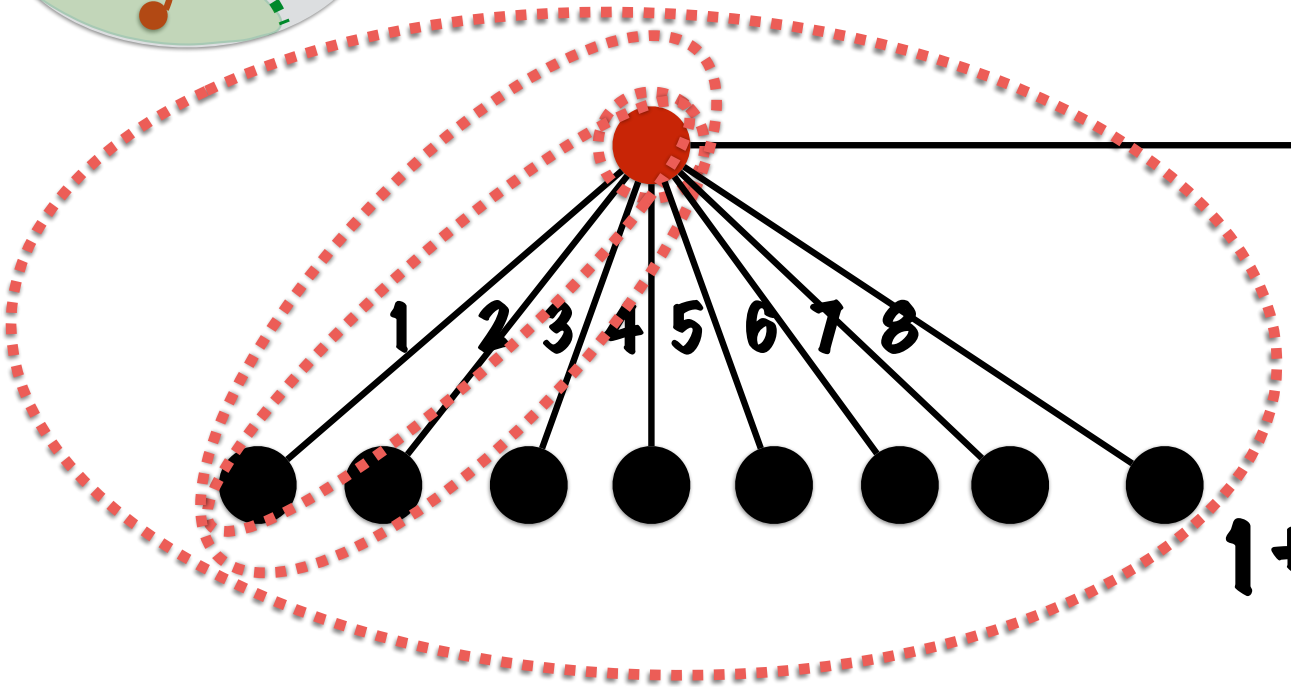
freeze y_S in tight constraints

Fact: final x,y are feasible.

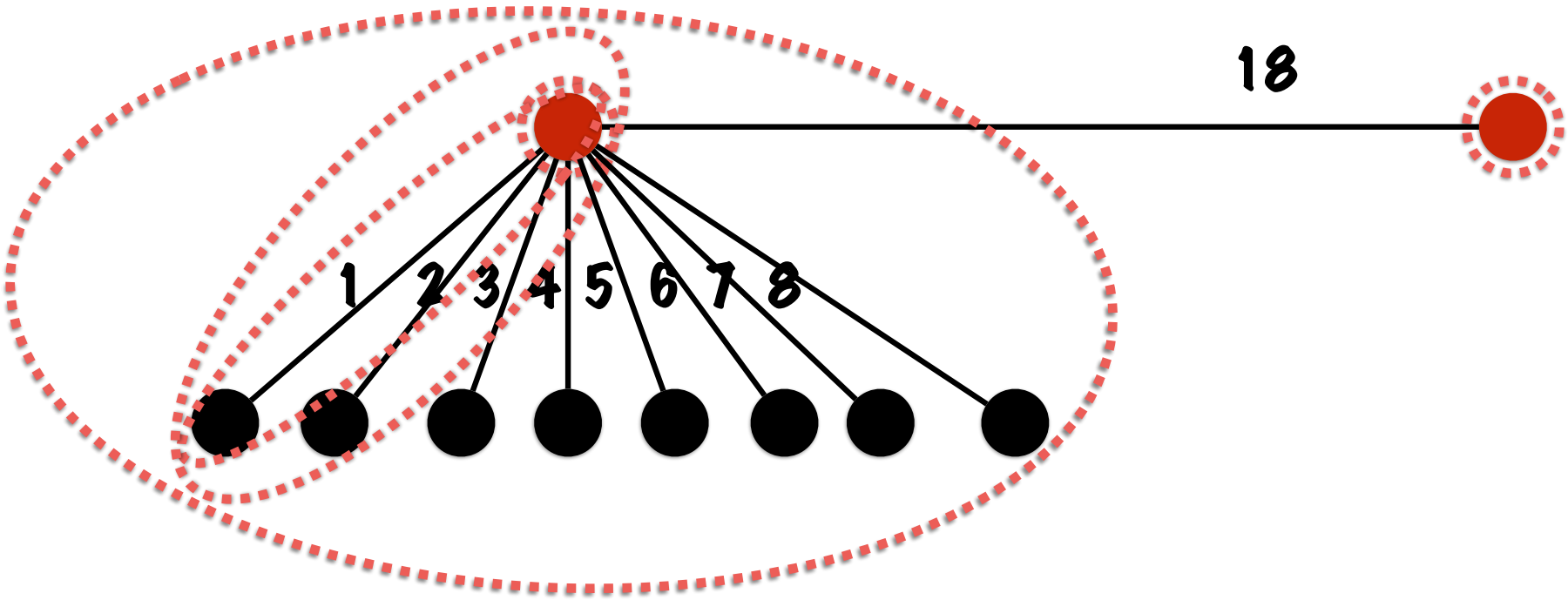
What about output cost?







Output cost
1+2+…+8+18
dual value
1+1+…+1+1+9=18
OPT
18
Bad!



Observe: Many of the edges in the output are useless

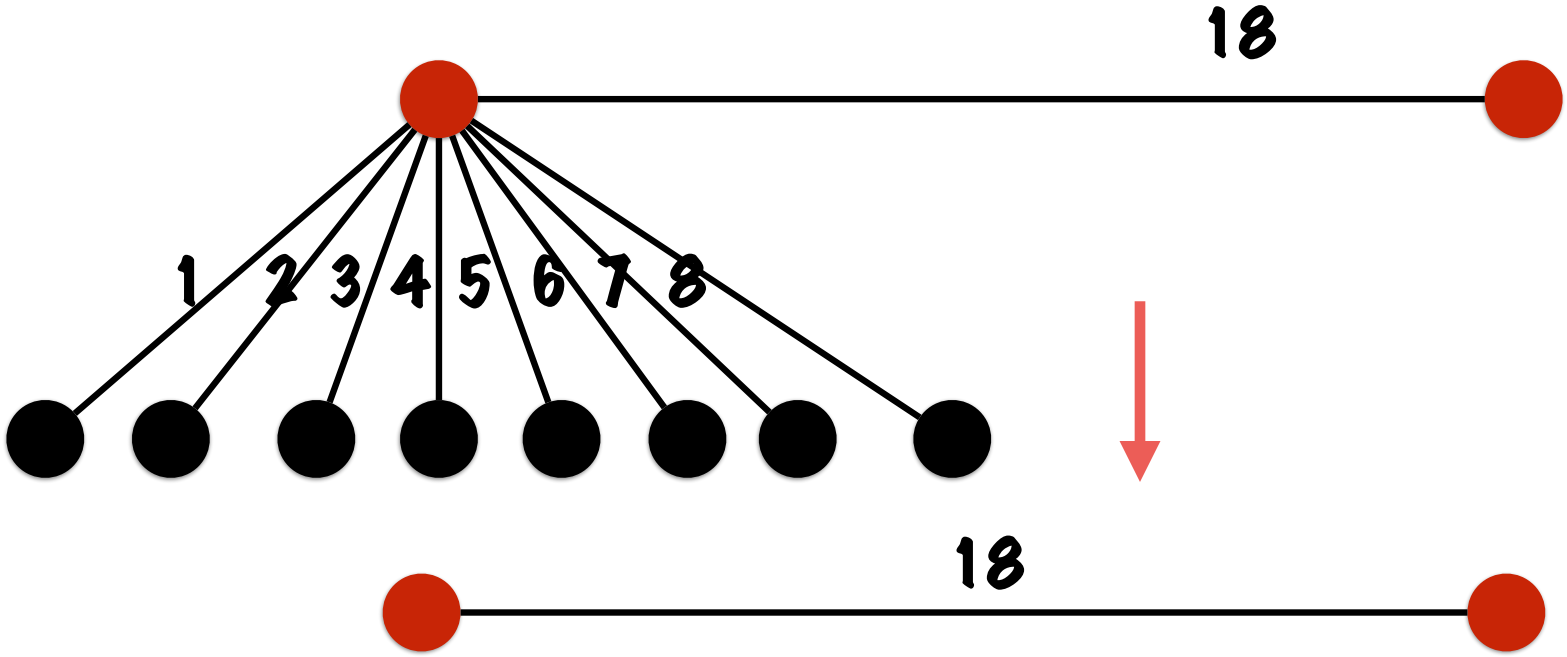
Idea: prune useless edges

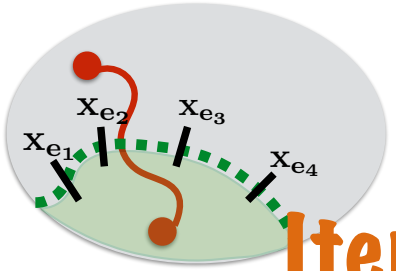
Modified algorithm

Consider set of edges defined by x

remove unnecessary edges

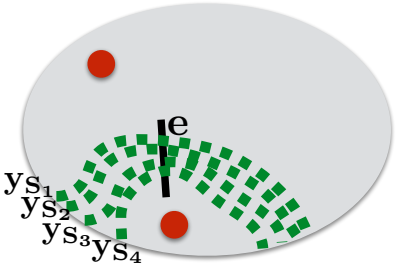
Output resulting set





Initialization:

$$x \leftarrow 0, y \leftarrow 0$$



Iteration: while x not satisfiable

in parallel, raise every unfrozen y_S with S minimal

stopped by tight constraint (e)

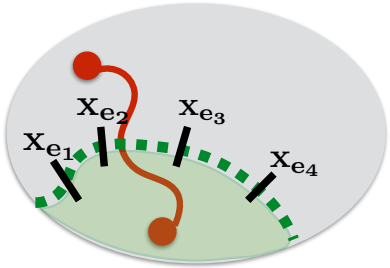
$$x_e \leftarrow 1$$

freeze y_S in tight constraints

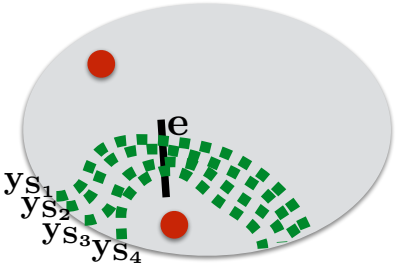
Pruning: let F={edges defined by x}

for each edge e of F in reverse order

remove e if unnecessary



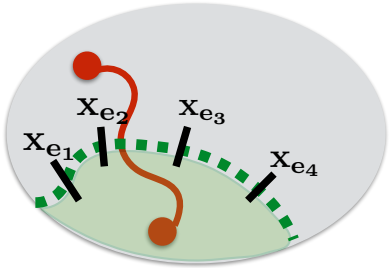
Theorem



It's a 2-approximation for Steiner forest

Steiner forest

Steiner forest



Initialization:

$x \leftarrow 0, y \leftarrow 0$

Iteration: while x not satisfiable

in parallel, raise every unfrozen y_S with S minimal

stopped by tight constraint (e)

$x_e \leftarrow 1$

freeze y_S in tight constraints

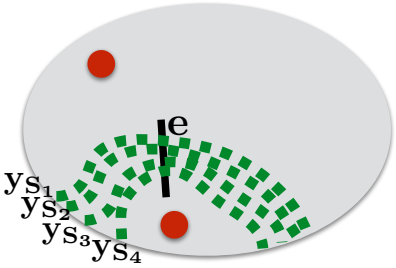
Pruning: let F={edges defined by x}

for each edge e of F in reverse order,

remove e if unnecessary

Theorem:

It's a 2-approximation for Steiner forest



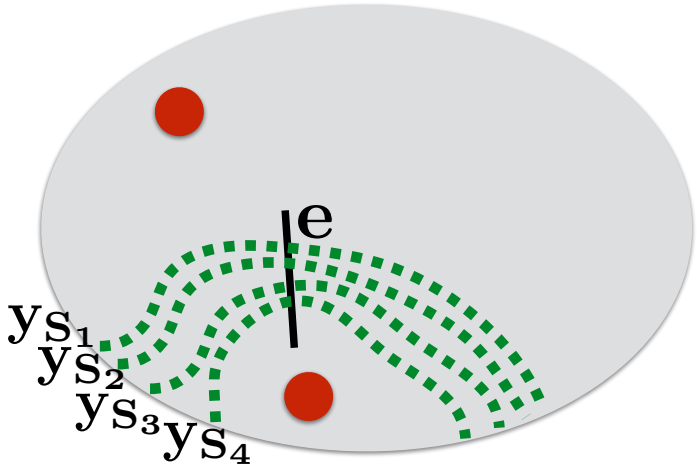
Observations:

- Final y is feasible
- Final x is a feasible forest
- Output F' is a forest
- its leaves are terminals
- slackness condition:

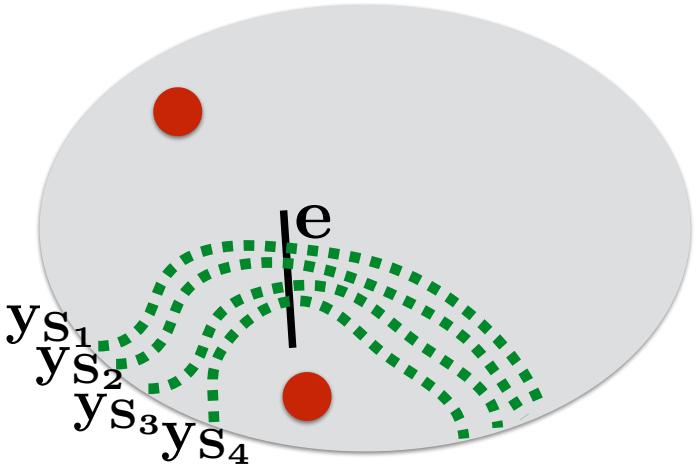
$$e \in F' \implies e \in F$$

$$\implies x_e = 1$$

$$\implies \sum_{S:e\in\delta(S)} y_S = c_e$$



slackness condition implies bound on cost of output by using dual variables:



$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S$$

As usual, invert summations:

$$\sum_{\mathbf{e}\in\mathbf{F}'} \sum_{\mathbf{S}:\mathbf{e}\in\delta(\mathbf{S})} \mathbf{y_S} = \sum_{\mathbf{S}} \mathbf{y_S} |\mathbf{F}' \cap \delta(\mathbf{S})|$$

Q: How to upper bound

$$\sum_{\mathbf{S}} \mathbf{y_S} |\mathbf{F}' \cap \delta(\mathbf{S})|$$

A: Incrementally. Bound

$$\sum_{\mathbf{time\ t}} \sum_{\mathbf{S}\ \text{“active” between } t \text{ and } t+dt} \epsilon |\mathbf{F}' \cap \delta(\mathbf{S})|$$

S “active” if its dual variable is being raised

Definition of "active" at current time

Initialization:

$x \leftarrow 0, y \leftarrow 0$

Iteration: while x not satisfiable

in parallel, raise every unfrozen y_S with S minimal

stopped by tight constraint (e)

$x_e \leftarrow 1$

freeze y_S in tight constraints

Pruning: let F={edges defined by x}

remove unnecessary edges from F

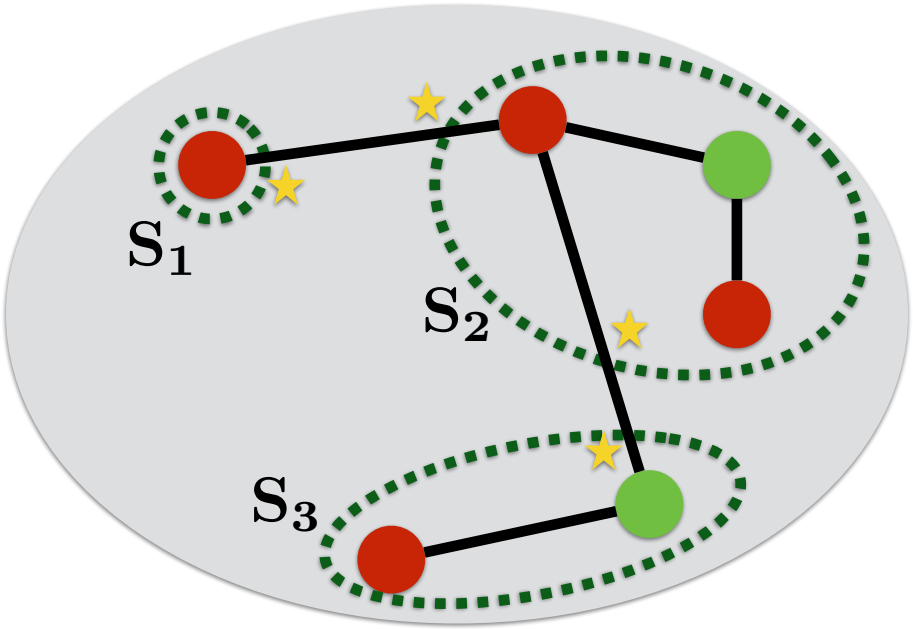
"active": unfrozen with S minimal

Q: How to upper bound

$$\sum_{\mathbf{S} \text{ currently active}} |\mathbf{F}' \cap \delta(\mathbf{S})|$$

A: Let's try some examples

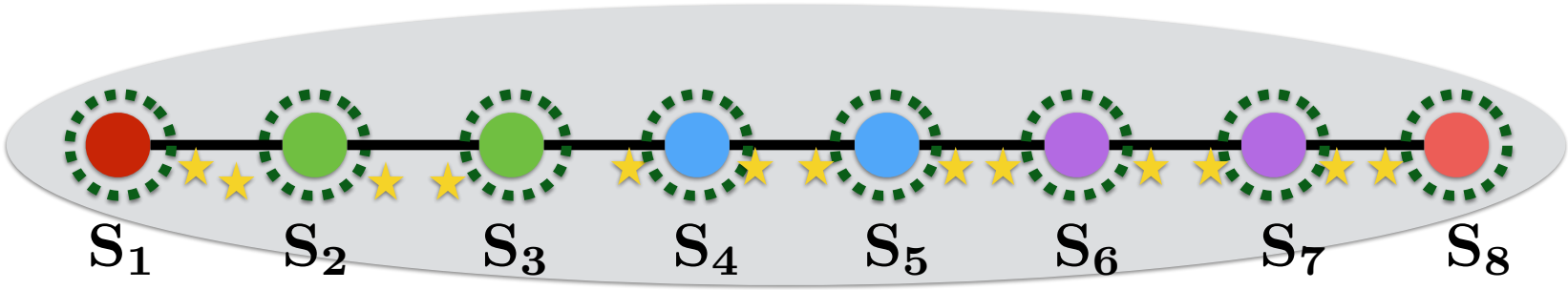
F'



3 active sets: S_1, S_2, S_3

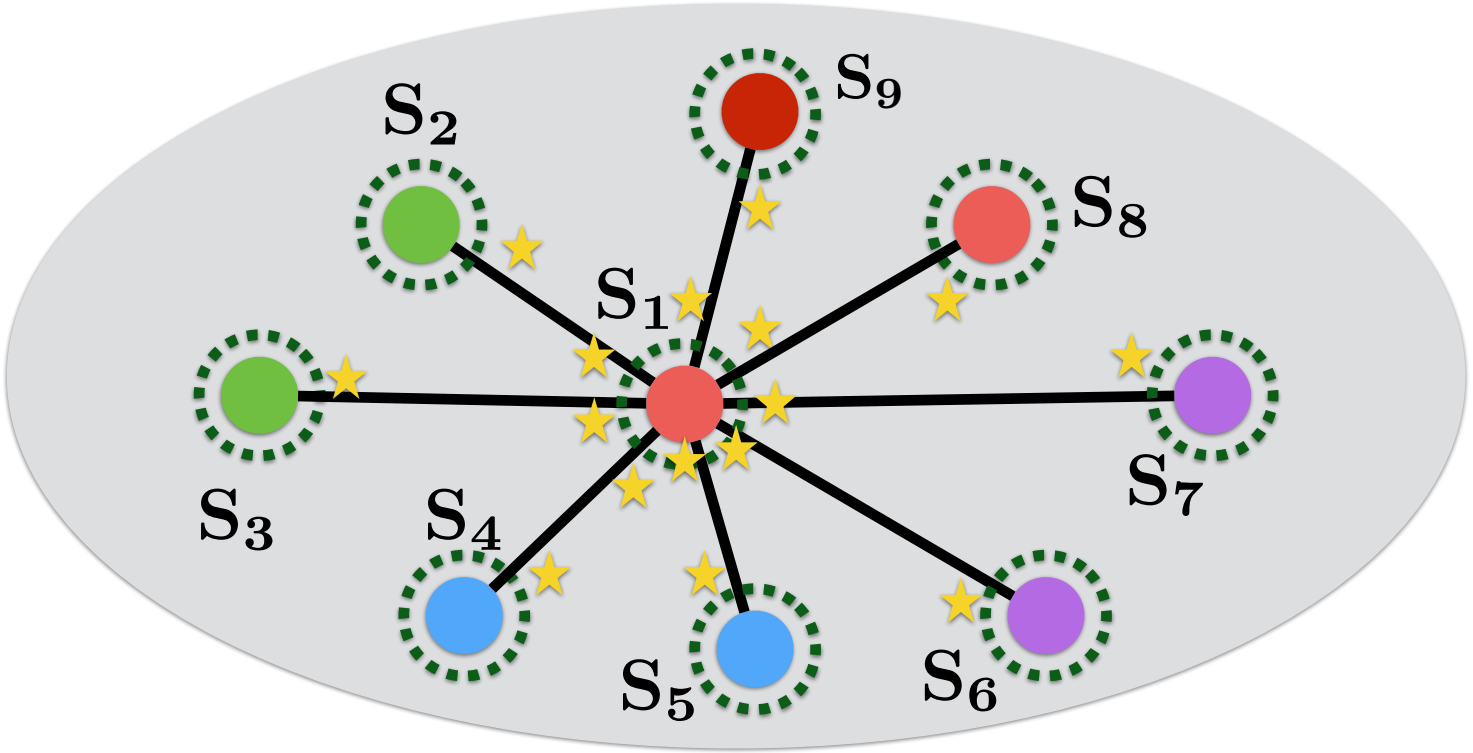
$$\sum_{S \text{ active}} |F' \cap \delta(S)| = \#(\text{stars}) = 1 + 2 + 1$$

$\overline{F'}$



8 active sets
14 stars

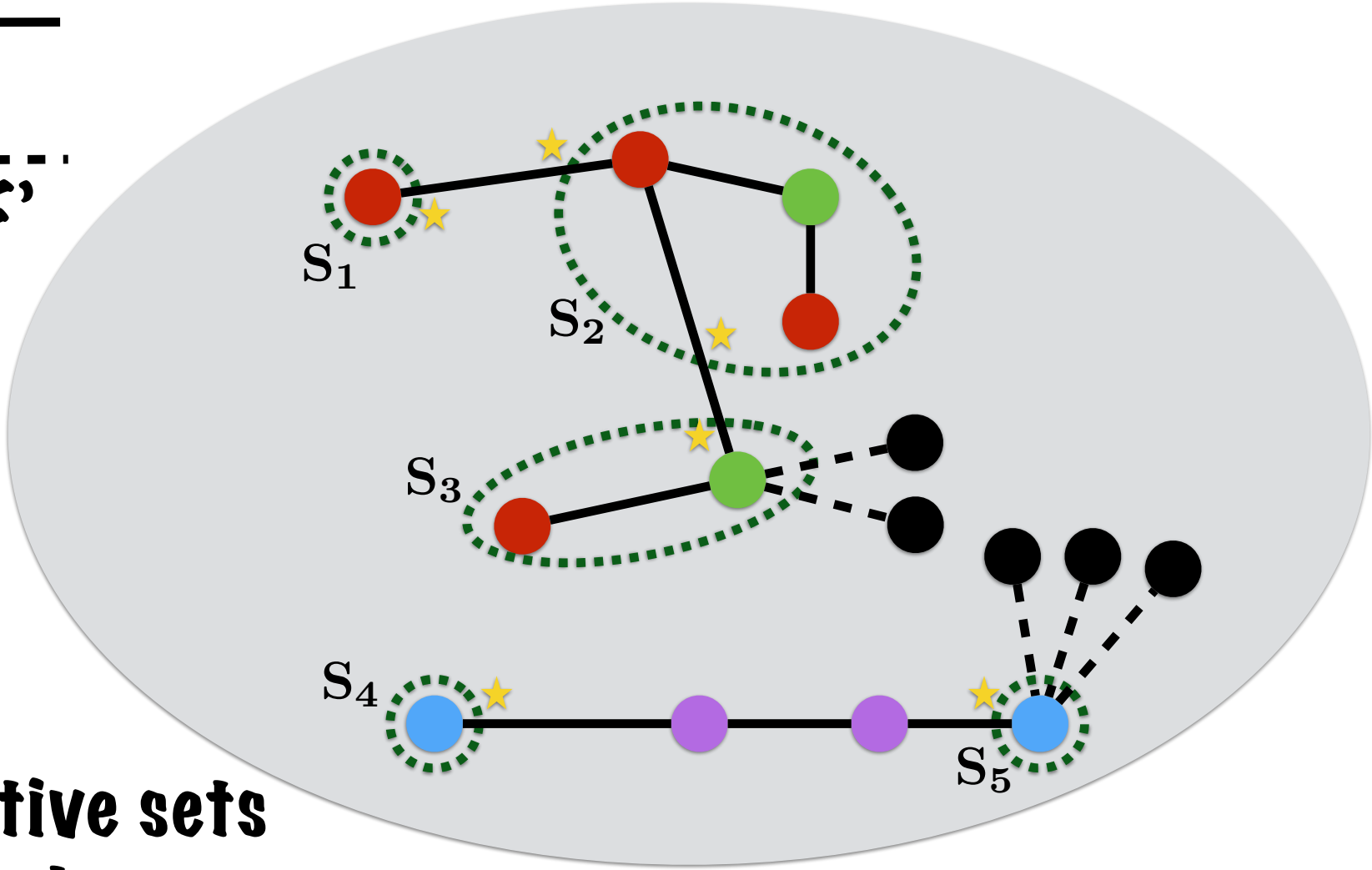
$\overline{F'}$



9 active sets
16 stars

F'

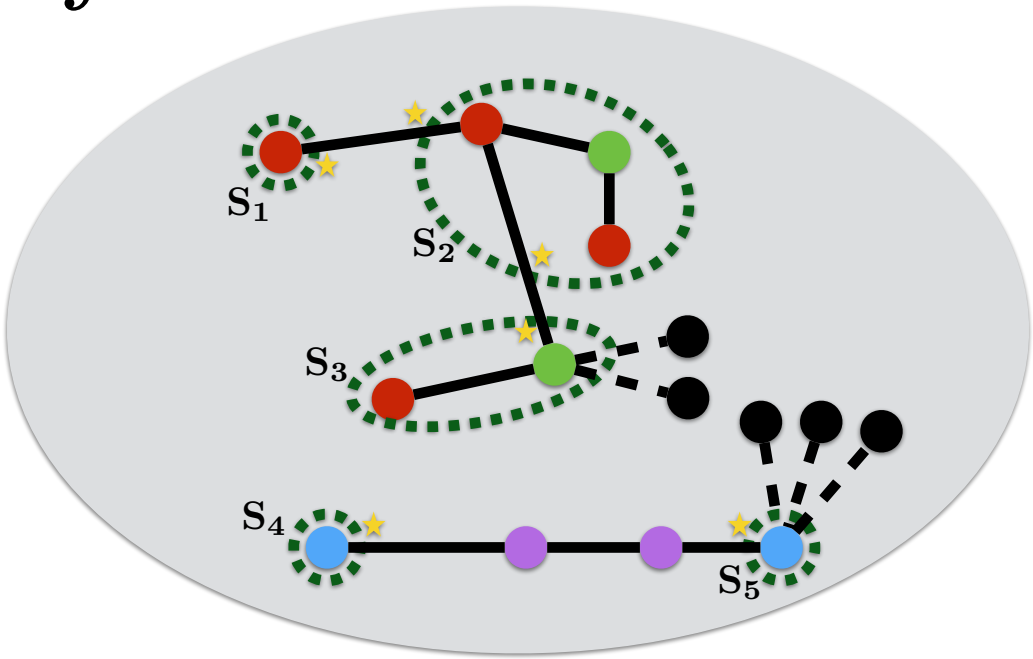
F-F'



5 active sets
6 stars

Q: How to upper bound

$$\sum_{S \text{ currently active}} |F' \cap \delta(S)|$$



Lemma: $$\frac{\sum_{S \text{ currently active}} |F' \cap \delta(S)|}{\#(\text{currently active sets})} \leq 2$$

Analysis assuming the lemma holds

Cost(output)=

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S$$

$$= \sum_S y_S |F' \cap \delta(S)|$$

$$= \sum_t \sum_{S \text{ active}} \epsilon |F' \cap \delta(S)|$$

$$\leq \sum_t 2\epsilon \cdot \#(S \text{ active})$$

$$= 2 \sum_S y_S$$

$$\leq 2 \cdot \text{OPT}$$

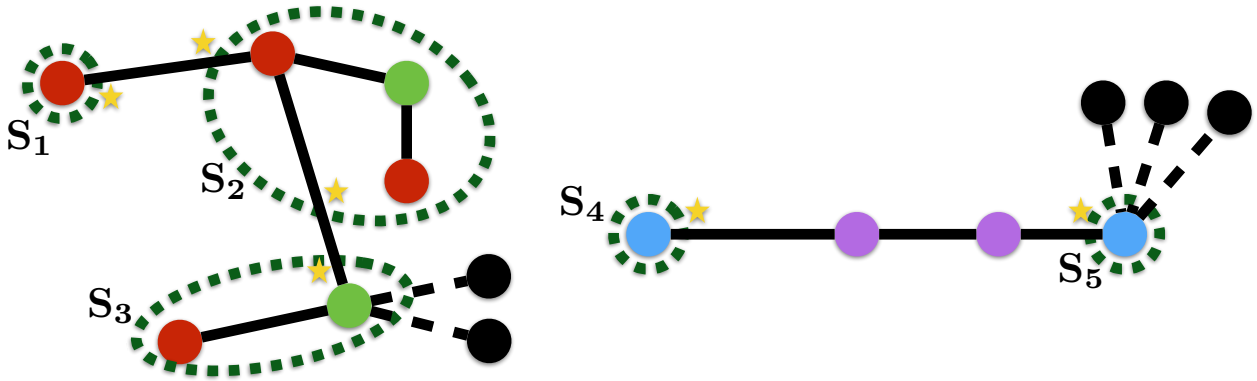
Steiner forest

Steiner forest

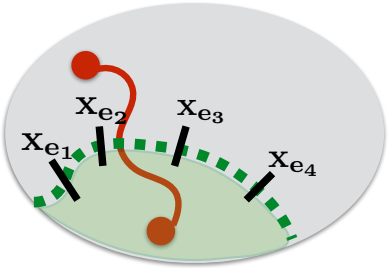
Lemma

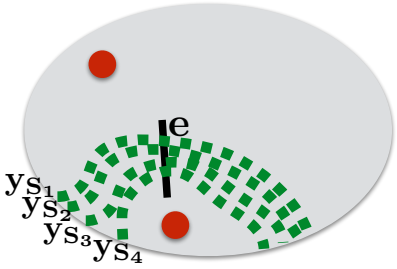
Let F' denote the output set of edges
Fix a time of the execution and say a set S is active if its dual variable is being raised.
Then:

$$\frac{\sum_{S \text{ active}} |F' \cap \delta(S)|}{\#(\text{active sets})} \leq 2$$



What properties do F' and active sets have?





Initialization:

$x \leftarrow 0, y \leftarrow 0$

Iteration: while x not satisfiable

in parallel, raise every unfrozen y_S with S minimal

stopped by tight constraint (e)

$x_e \leftarrow 1$

freeze y_S in tight constraints

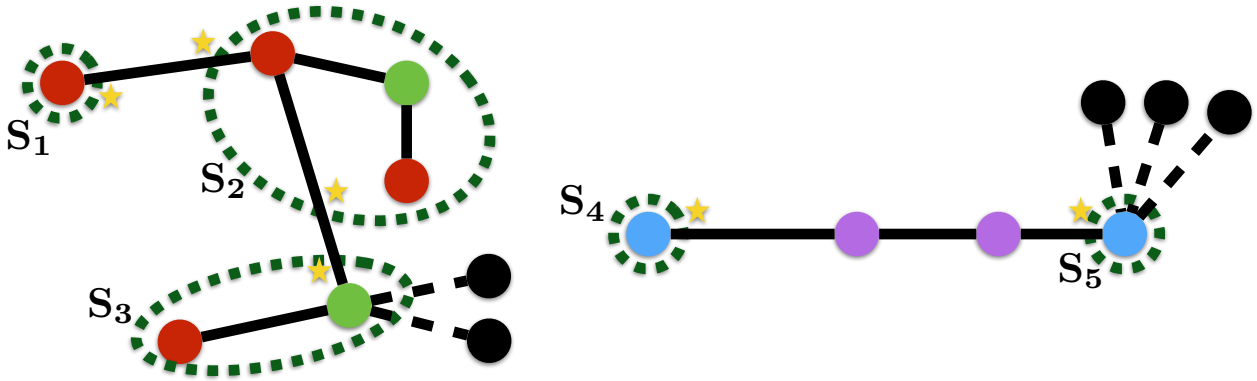
Pruning: let F={edges defined by x}

for each edge e of F in reverse order,

remove e if unnecessary

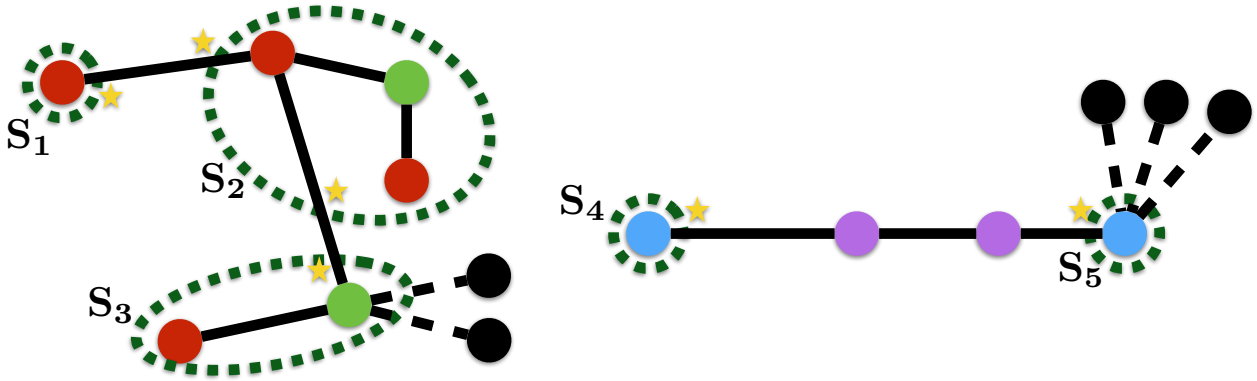
Properties of active sets

active sets are disjoint subsets of nodes containing terminals



Properties of F'

F' is a forest
and its leaves are terminals

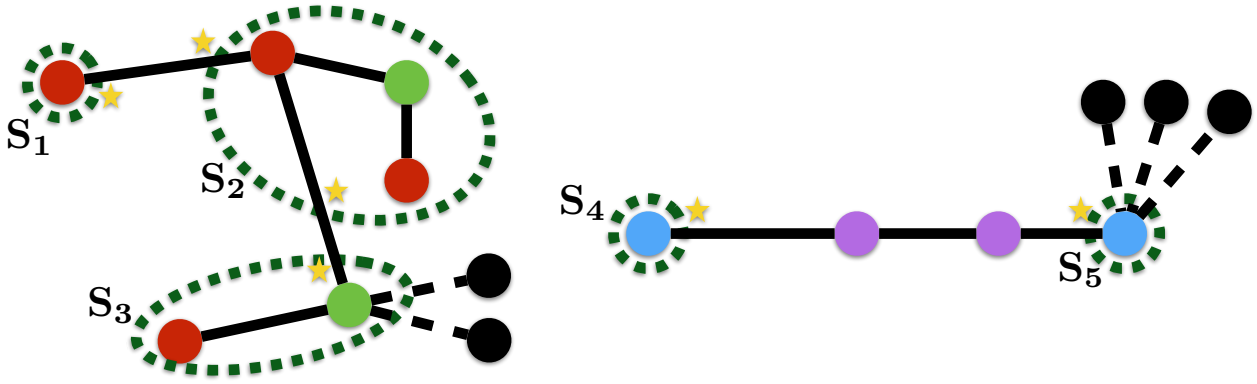


Joint properties of F' and of active sets

Consider a tree of F'

Assume it intersects some active sets

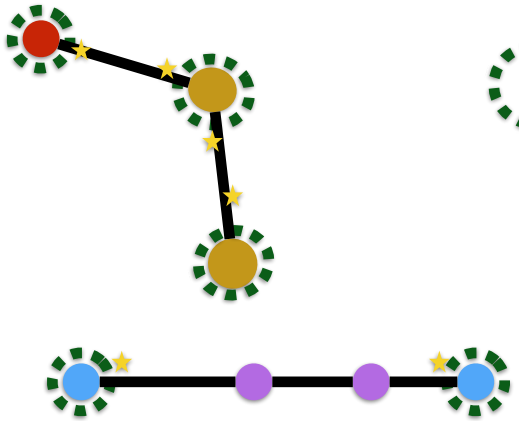
- the active sets are subtrees
- all the leaves are in active sets



Graph theory lemma

Given a tree T and a subset X of its vertices that includes all the leaves:

$$\sum_{v \in X} \deg(v) \leq 2 \cdot |X|$$

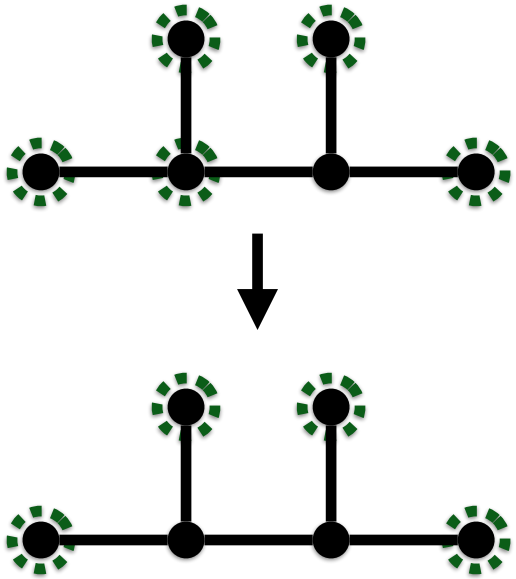


elements of X

Graph theory lemma

T tree, X subset of vertices including all leaves:

$$\sum_{v \in X} \deg(v) \leq 2 \cdot |X|$$



Proof: induction

- true for X=V

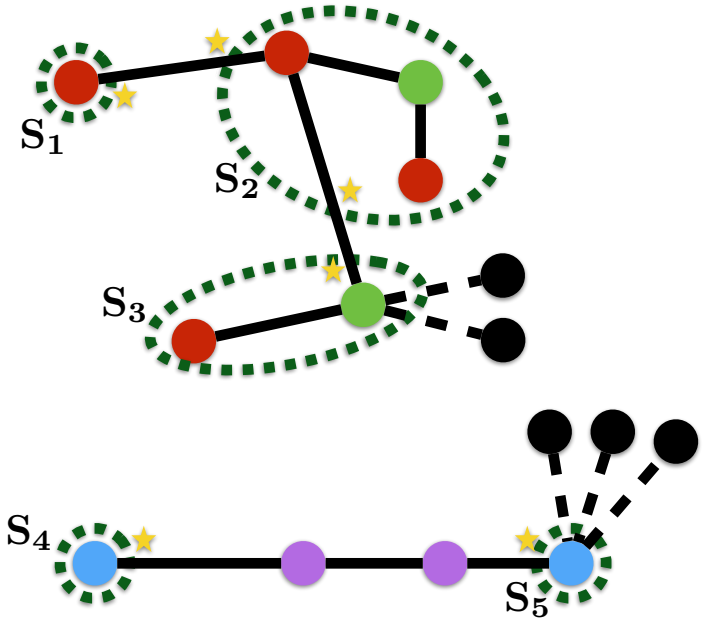
$$\sum_{v \in V} \deg(v) = 2|E| = 2(|V| - 1) < 2 \cdot |V|$$

- if true for X then remains true when removing an internal node from X, QED.

Lemma

F' output forest

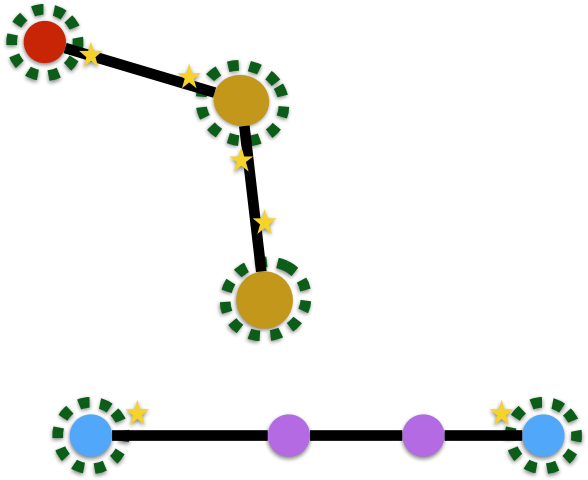
$$\frac{\sum_{S \text{ active}} |F' \cap \delta(S)|}{\#(\text{active sets})} \leq 2$$



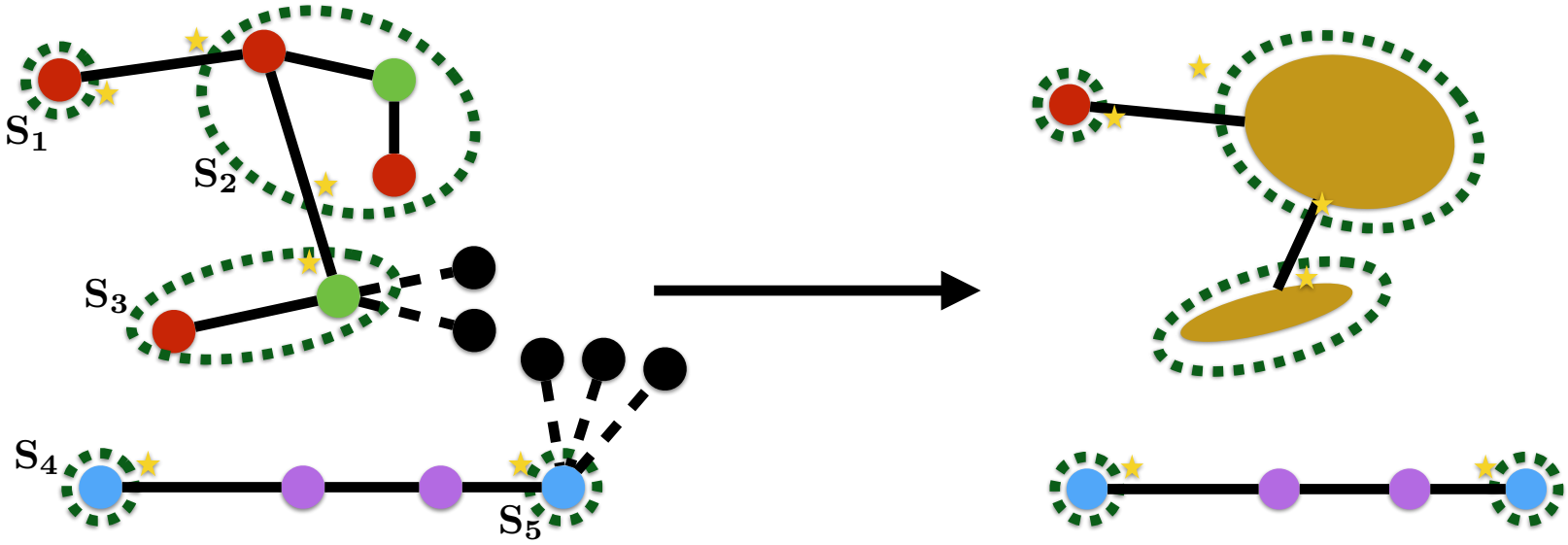
Graph theory lemma

T tree, X subset of vertices including all leaves:

$$\frac{\sum_{v \in X} \deg(v)}{|X|} \leq 2$$



Lemma F' output forest: $\frac{\sum_{S \text{ active}} |F' \cap \delta(S)|}{\#(\text{active sets})} \leq 2$



Contracting active sets does not change

$\sum_{S \text{ active}} |F' \cap \delta(S)|$ **nor** $\#(\text{active sets})$

QED

Result:

a 2-approximation algorithm
for the Steiner forest problem

Comments

- extends: connectivity problems, and beyond
- primal-dual is greedy
  - dual gives insight
- post-processing can be necessary
- combinatorial: LPs are guides, not computational tools

Steiner forest