Primal-Dual Algorithms

A schizophrenic lecture

Marek Adamczyk, Stefano Leonardi,
Erik Jan van Leeuwen

Recap

- Approximation algorithms
  - Get within factor of optimum solution
  - Example: FACILITY LOCATION: log n approximation in polynomial time
- Linear Programming

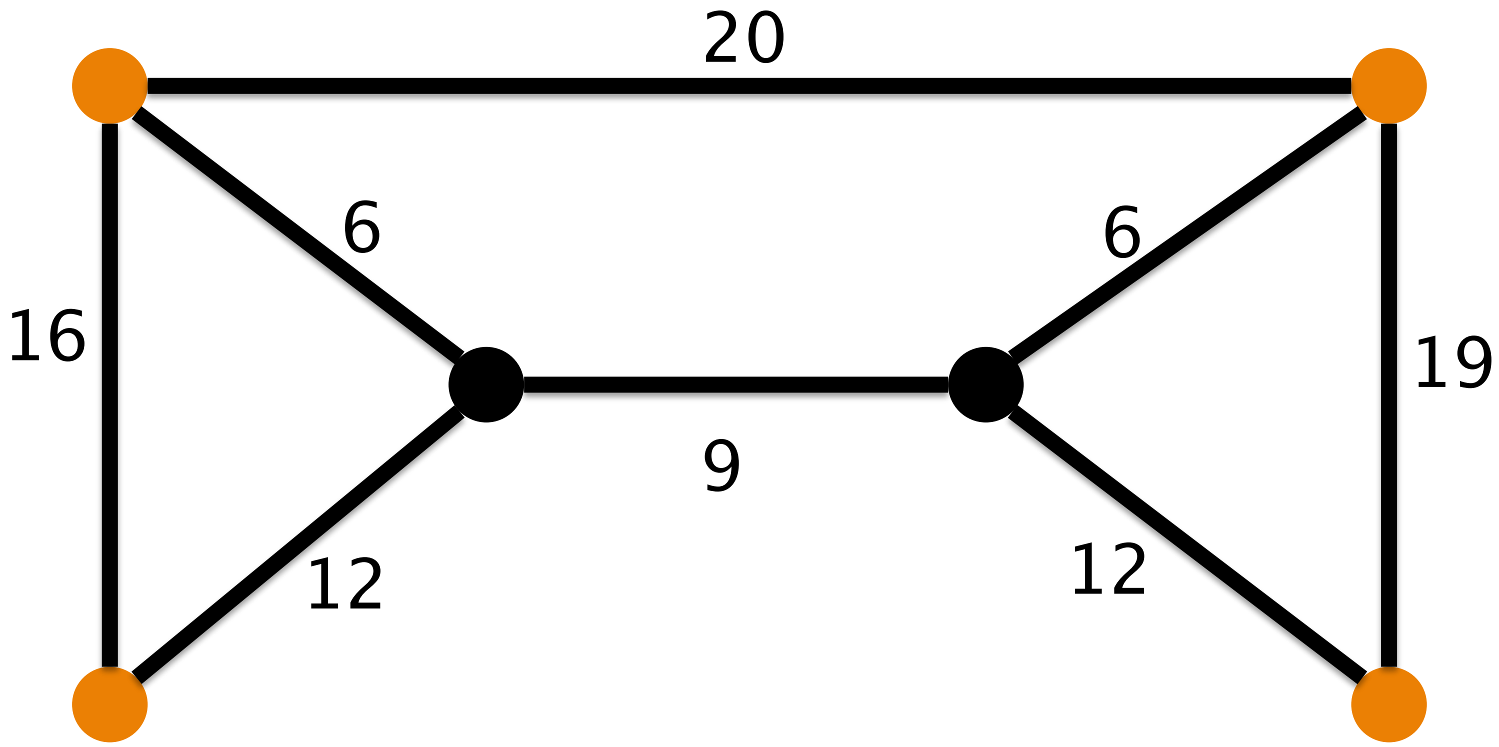
Today

- Combine linear programming and approximation algorithms
- Combine primal and dual of LP
- Problems: Steiner Tree and Metric Facility Location

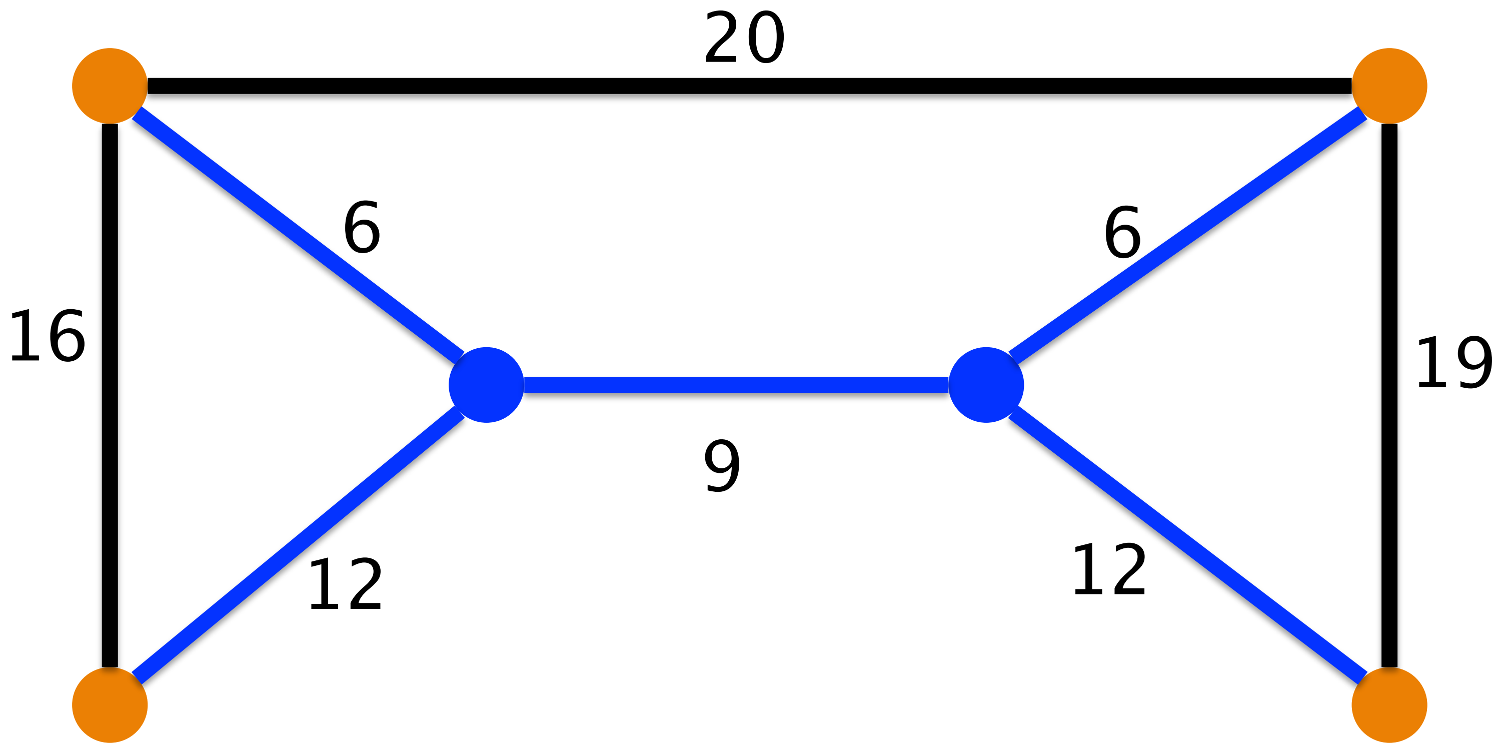
Steiner Tree

- Given:
  - Graph G, cost function c over E(G)
  - Set of terminals R
- Wanted: tree T of minimum total cost that contains all terminals

Steiner Tree



Steiner Tree



Metric and non-metric

- Metric: G is complete graph, $c(uv) \leq c(ux) + c(xv)$

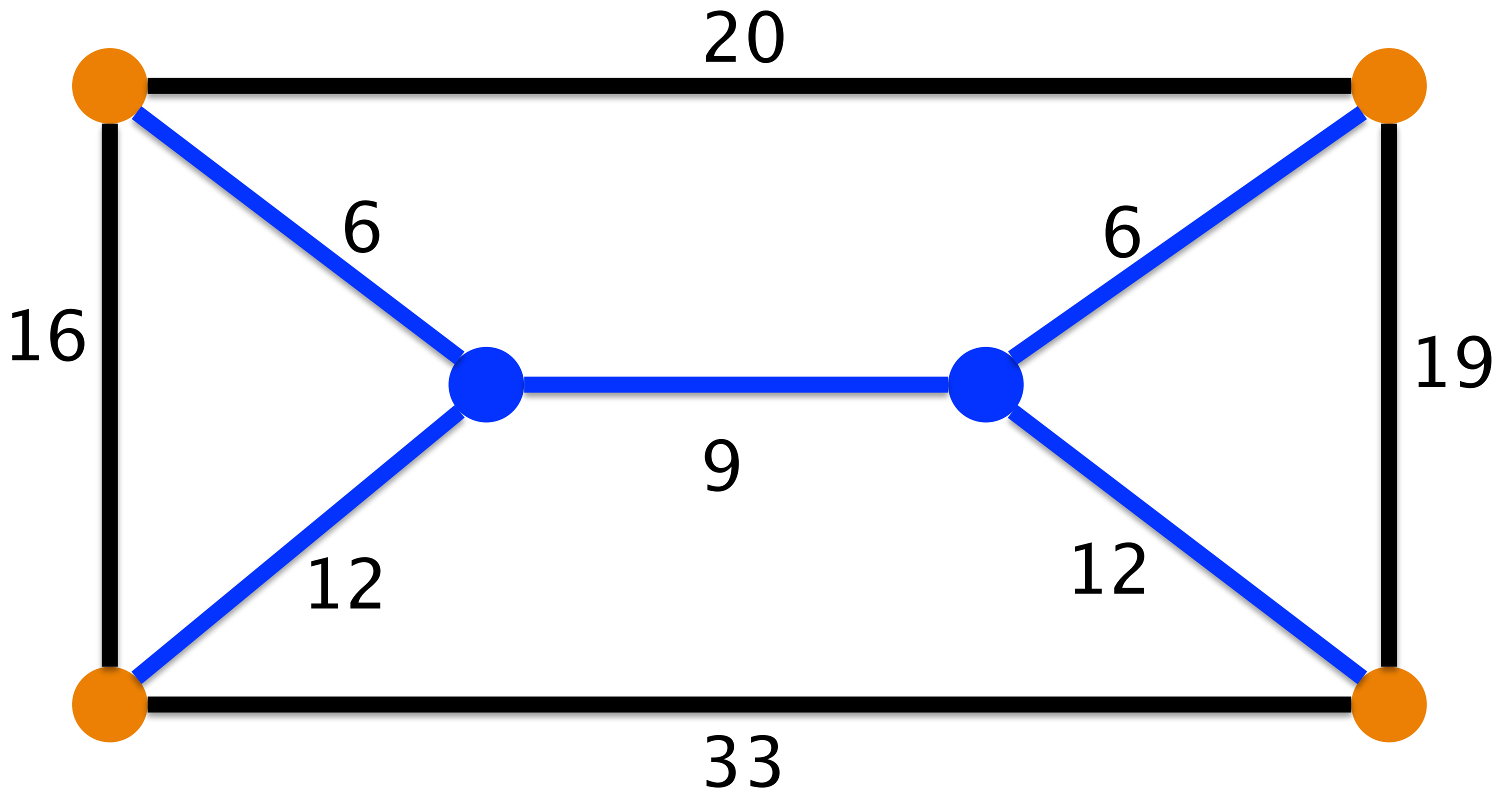
- There is reduction from non-metric to metric case

  - Use **metric closure**: set $c(uv) =$ cost of min-cost u-v path

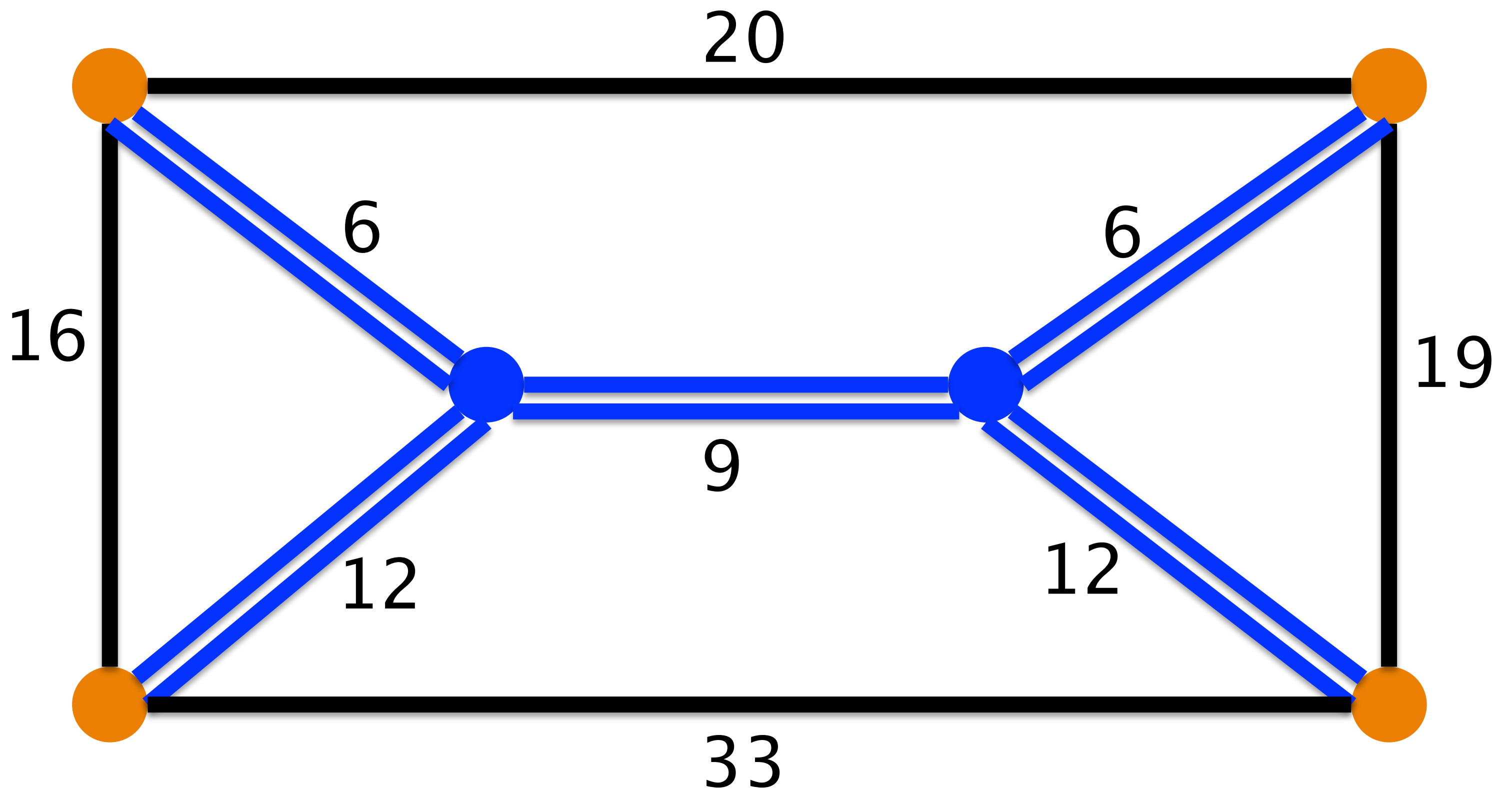
Algorithm

- Find a minimum spanning tree T of G[R]
- This gives solution in poly-time
- But how good/bad?
  - Transform optimum into spanning tree

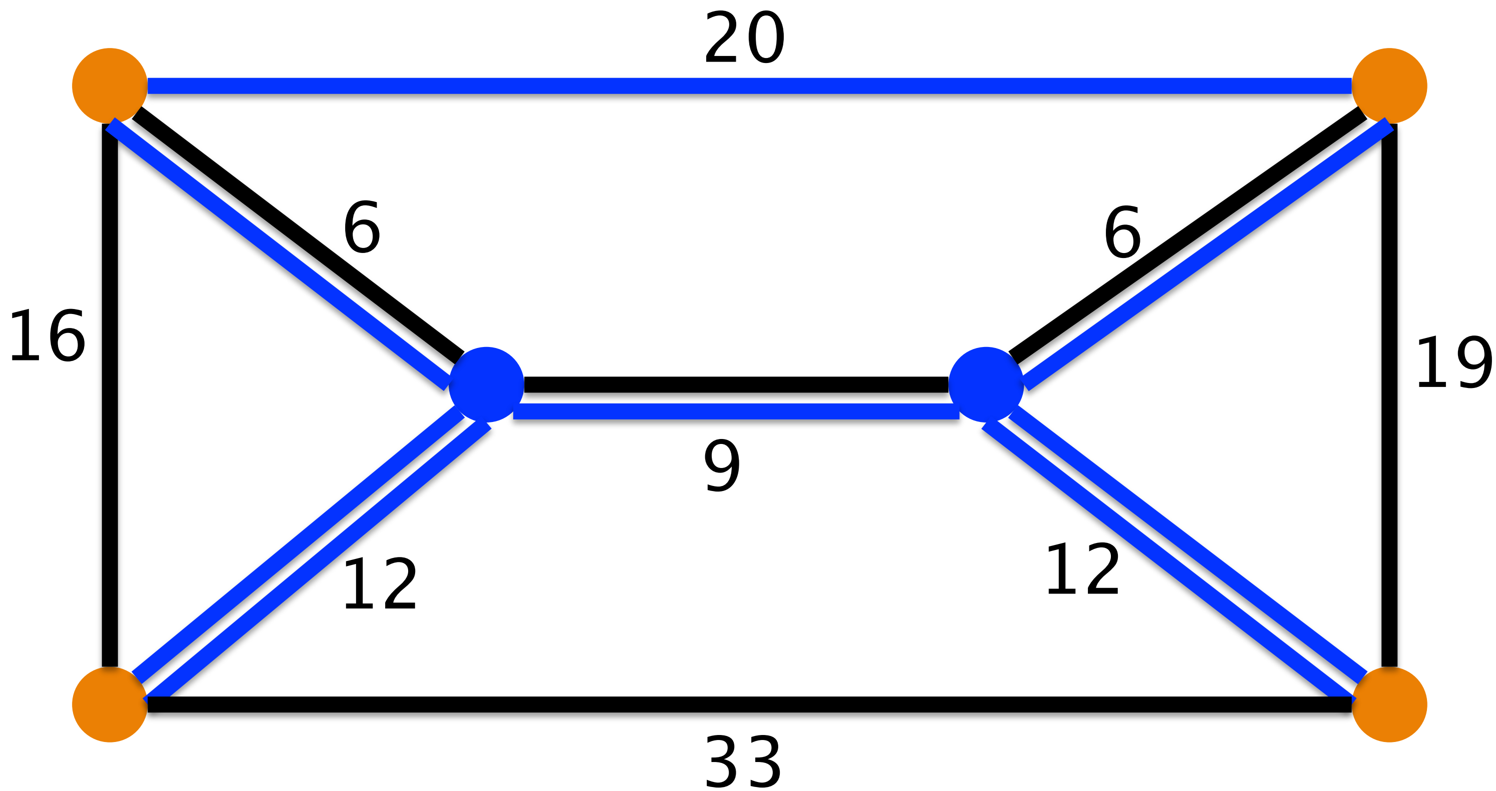
Example



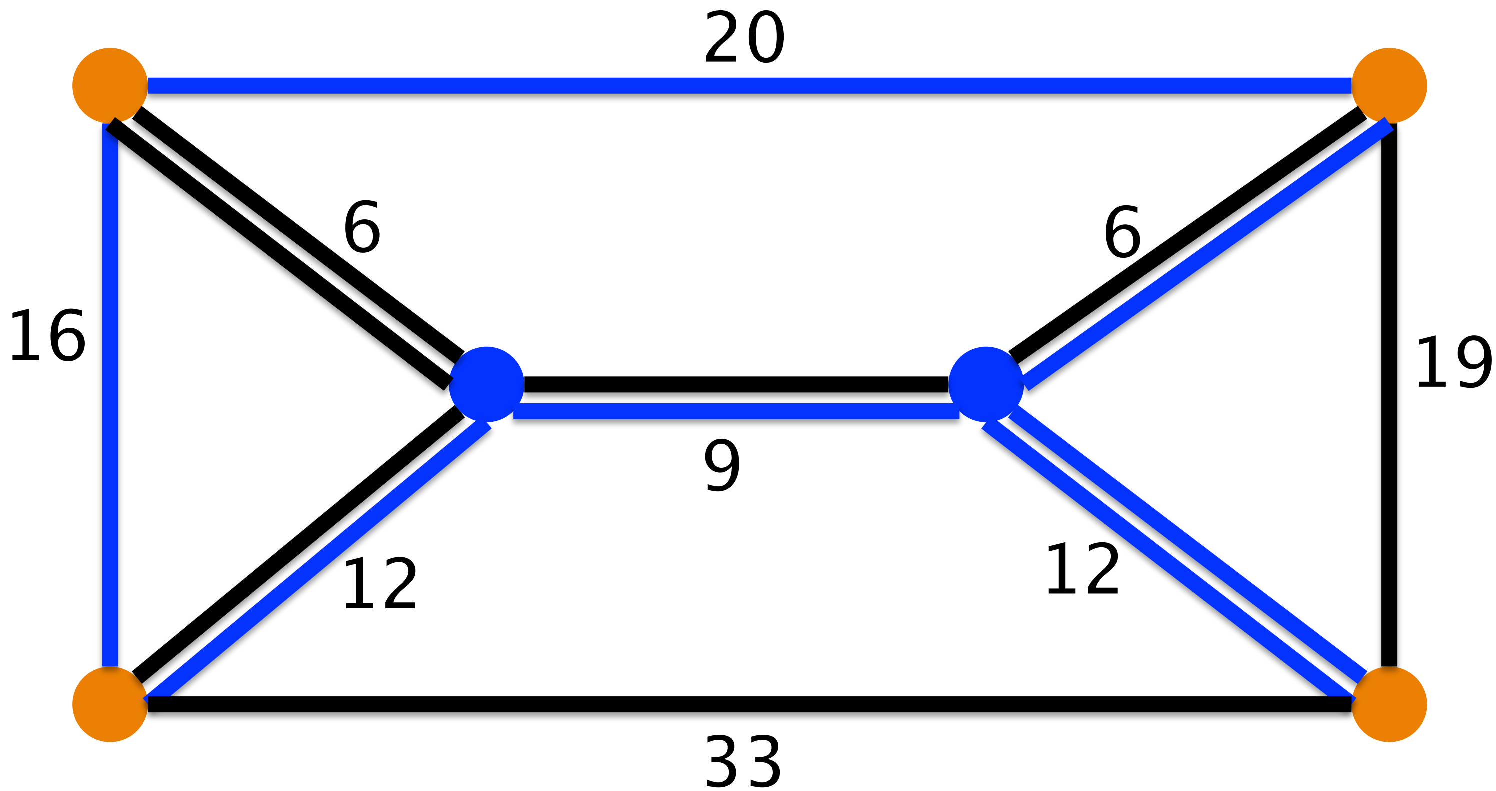
Example



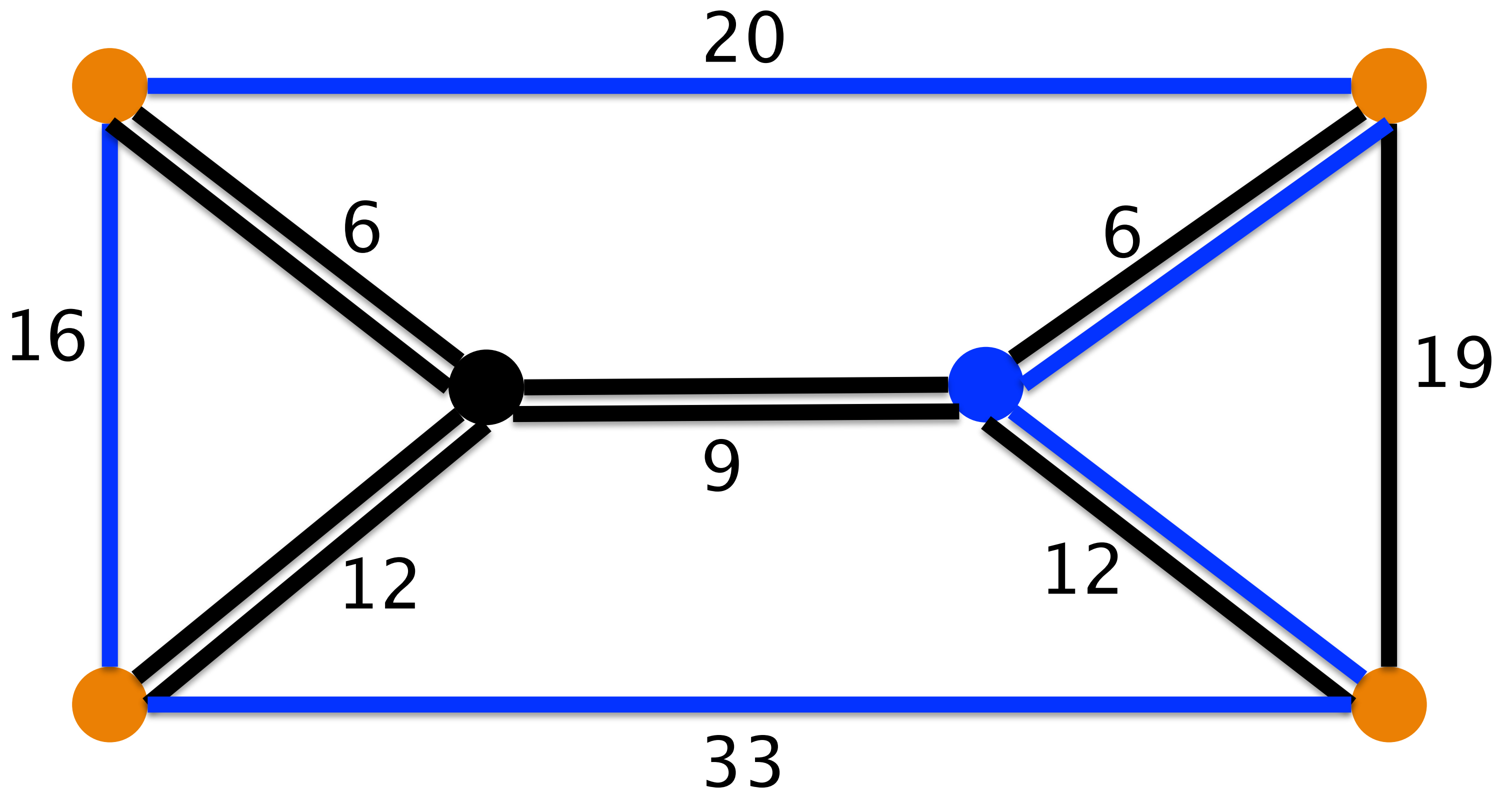
Example



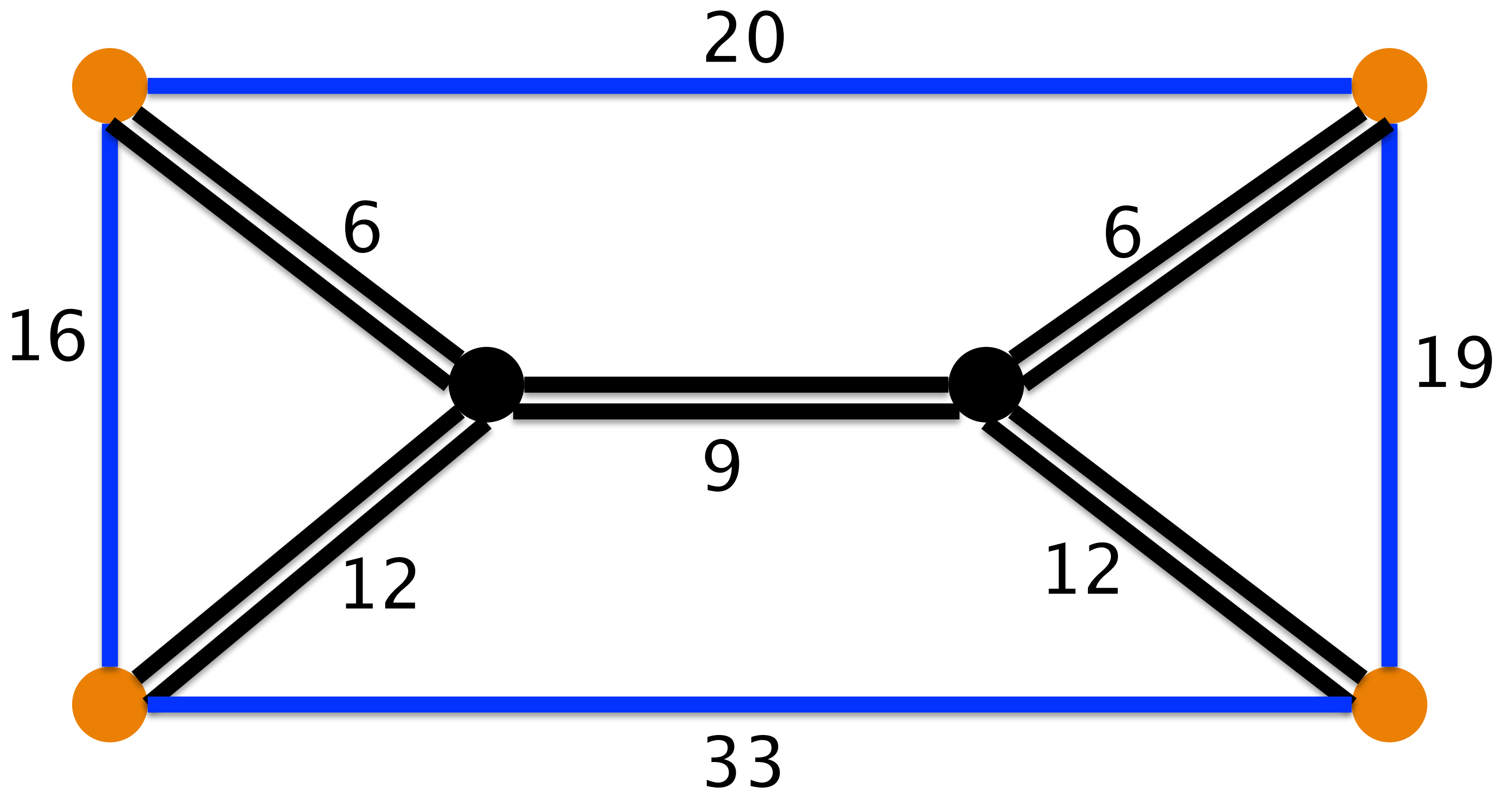
Example



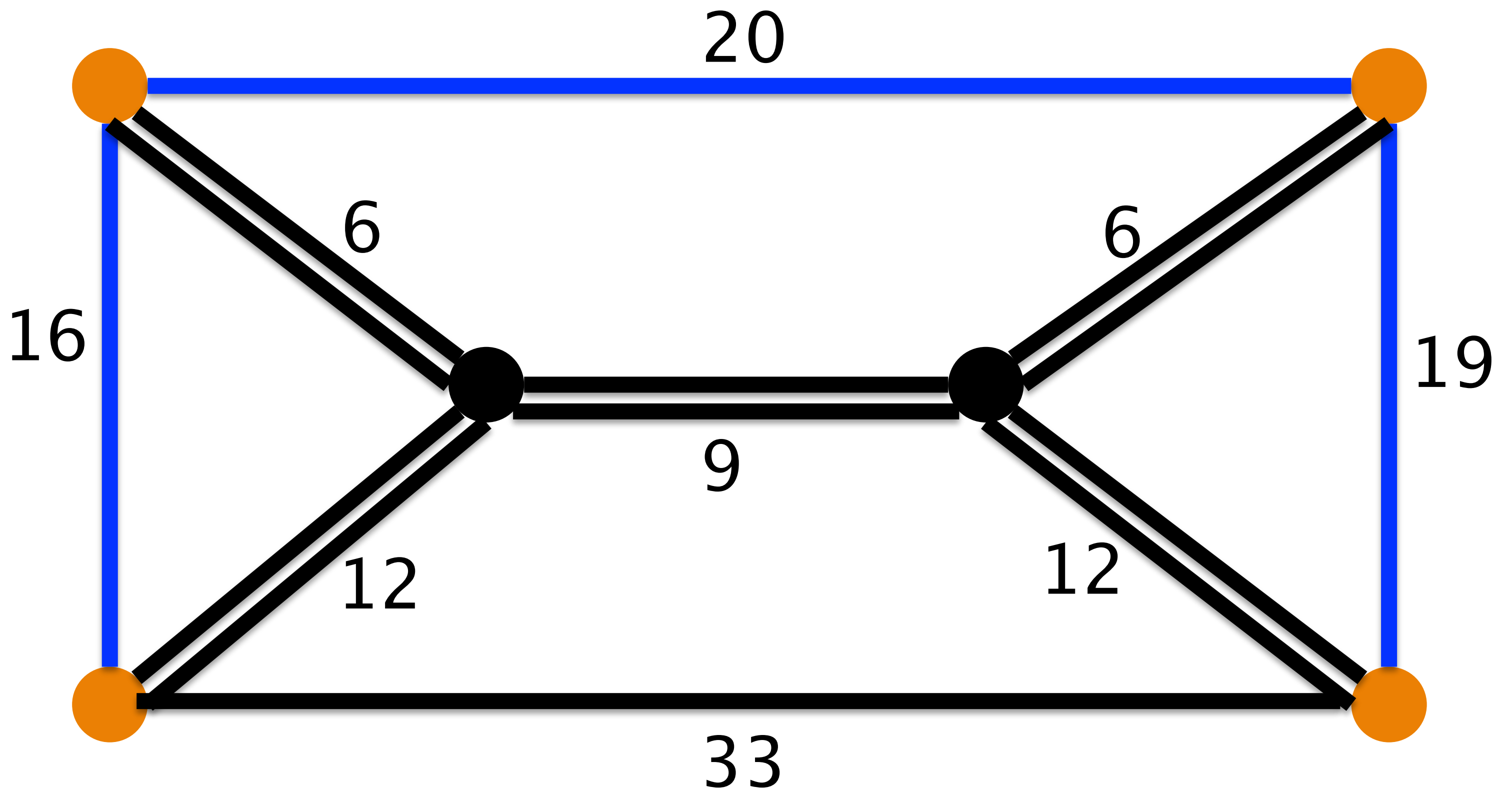
Example



Example



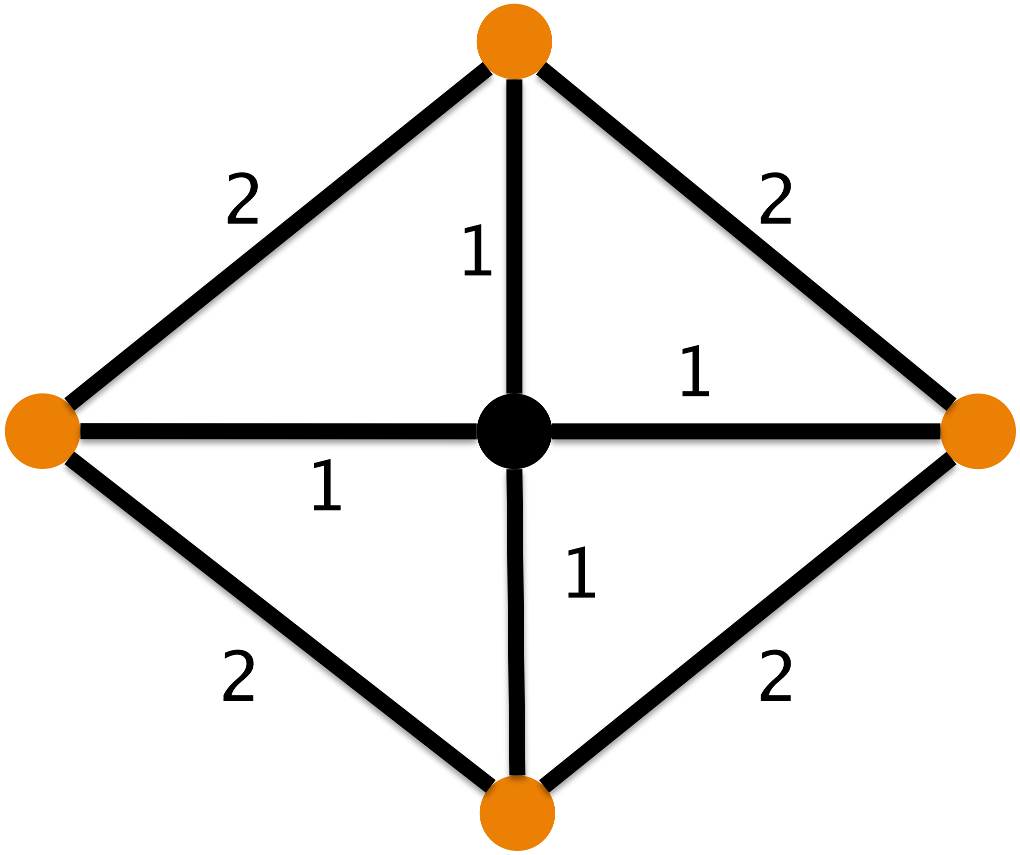
Example



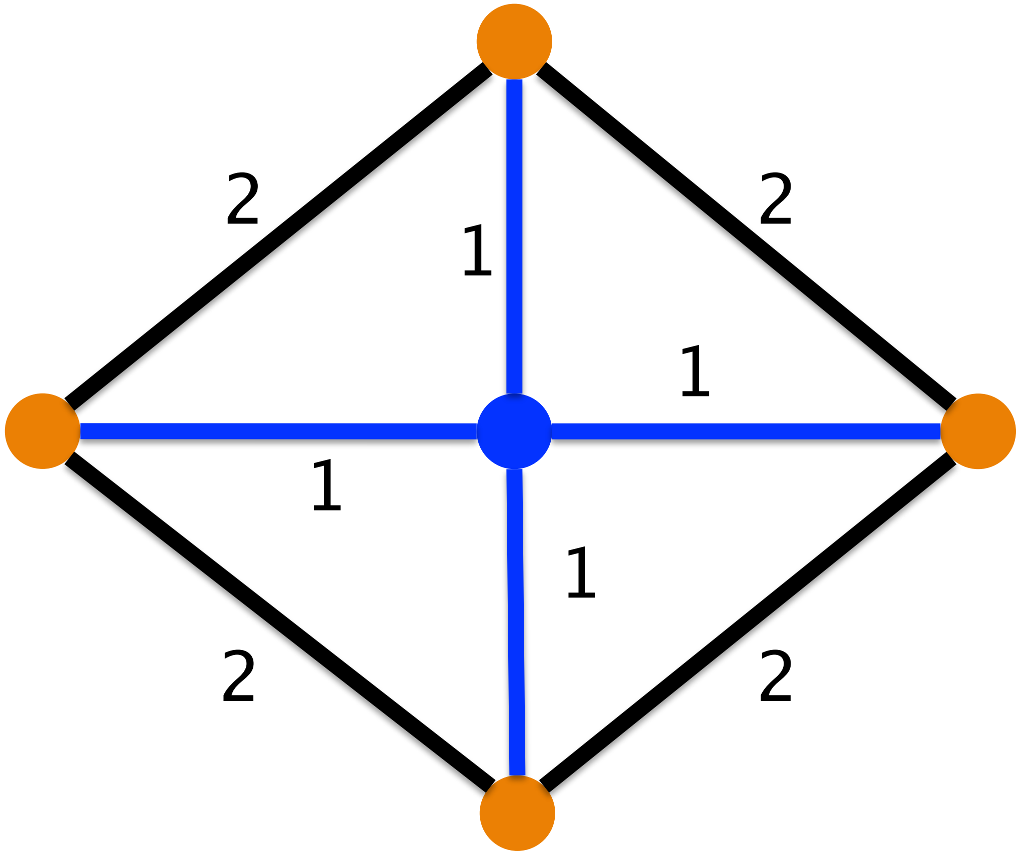
Analysis

- From optimum T^*, can obtain spanning tree T' of $G[R]$ such that $c(T') \leq 2\, c(T^*)$
- Since T is minimum spanning tree of $G[R]$, $c(T) \leq c(T') \leq 2\, c(T^*)$
- **2-approximation in poly-time**

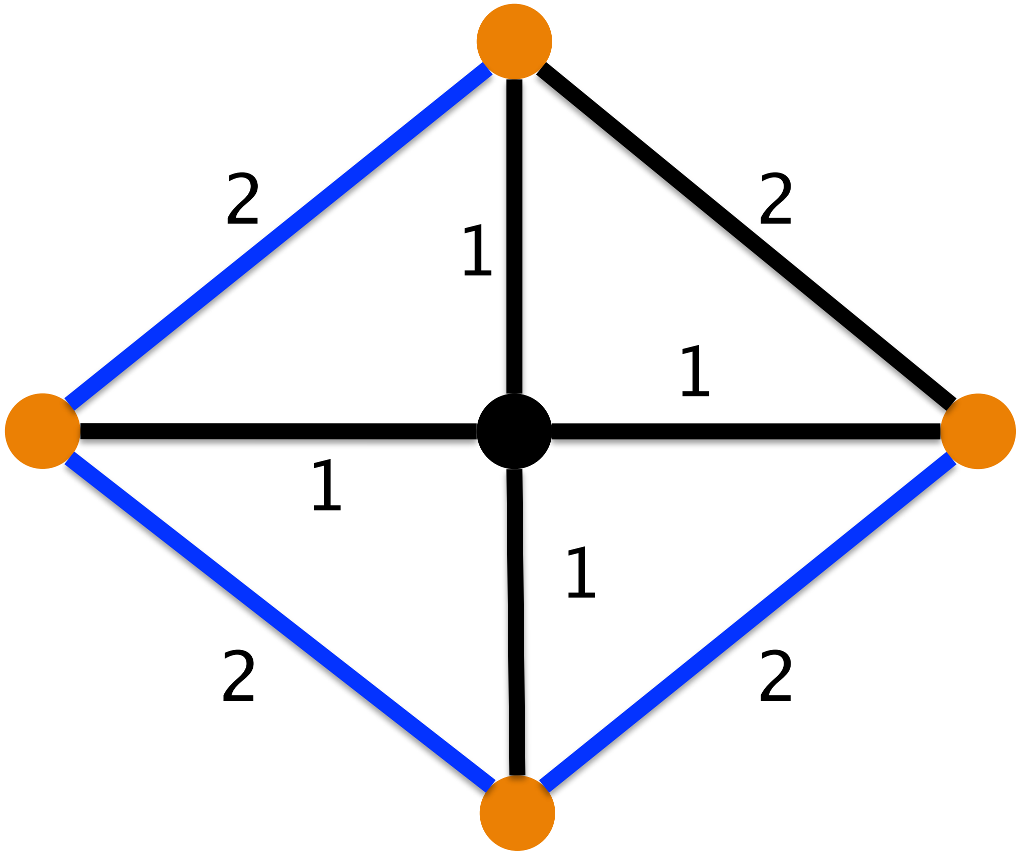
Can it be better?



Can it be better?



Can it be better?



Can it be better?

- Optimum has cost $|R|$
- Spanning tree has cost $\geq 2|R| - 2$
- Analysis is **tight**!

An LP perspective

A new view of STEINER TREE

LP formulation

- x_e: edge e in solution
- f(S) for any subset S of V(G): indicates terminal in S and V(G)–S
  - Edge must cross such **cut S**

LP formulation

$$\text{minimize} \quad \sum_{e \in E(G)} c_e x_e$$

$$\text{such that} \quad \sum_{e : e \in \delta(S)} x_e \geq f(S) \qquad \forall S \subseteq V(G)$$

$$x_e \in \{0, 1\} \qquad \forall e \in E(G)$$

LP relaxation

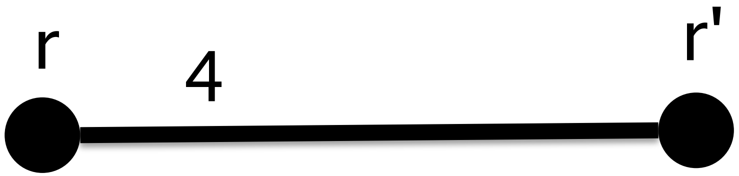
$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{e \in E(G)} c_e x_e & \\
\text{such that} & \displaystyle\sum_{e : e \in \delta(S)} x_e \geq f(S) & \forall S \subseteq V(G) \\
& x_e \geq 0 & \forall e \in E(G)
\end{array}
$$

LP dual

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{S \subseteq V(G)} f(S) \cdot y_S & \\
\text{such that} & \displaystyle\sum_{S : e \in \delta(S)} y_S \leq c_e & \forall e \in E(G) \\
 & y_S \geq 0 & \forall S \subseteq V(G)
\end{array}
$$

Dual in pictures

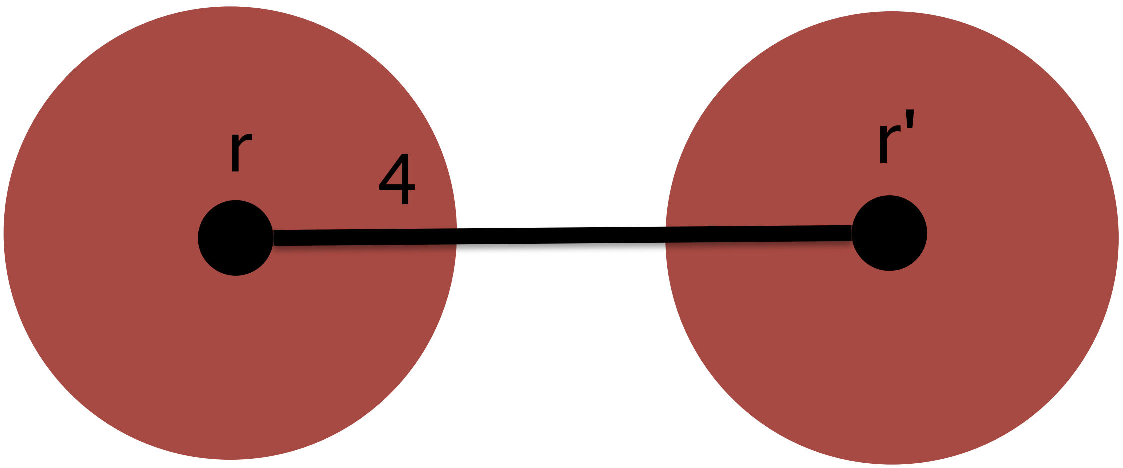
- y_S is disk around S with radius y_S



- Look at $y_{\{r\}}$ and $y_{\{r'\}}$

Dual in pictures

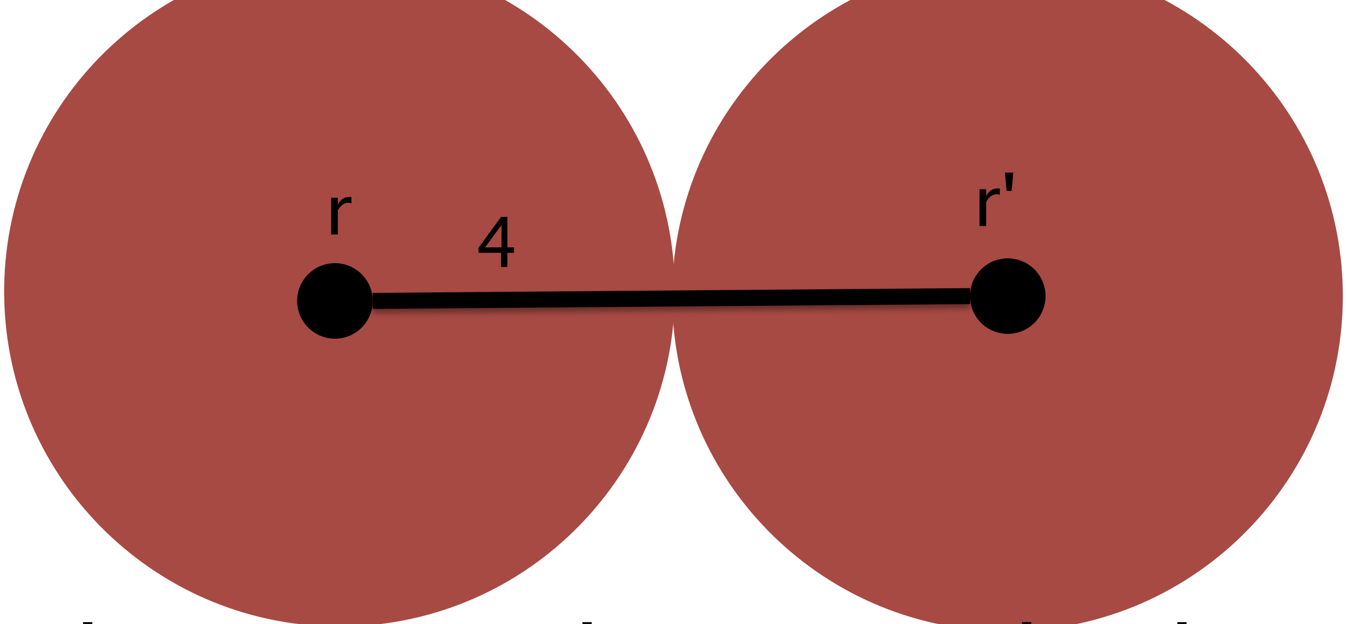
- y_S is disk around S with radius y_S



- Look at $y_{\{r\}}$ and $y_{\{r'\}}$; say both are 1

Dual in pictures

- y_S is disk around S with radius y_S

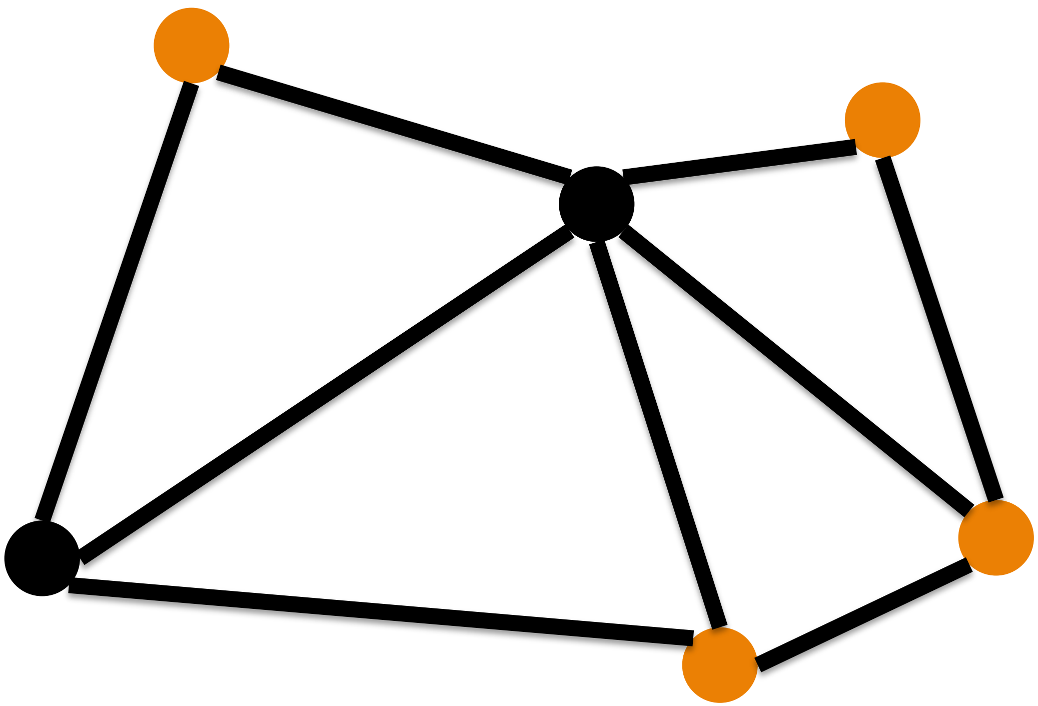


- Look at $y_{\{r\}}$ and $y_{\{r'\}}$; say both are 2
  (r,r') is called **tight**

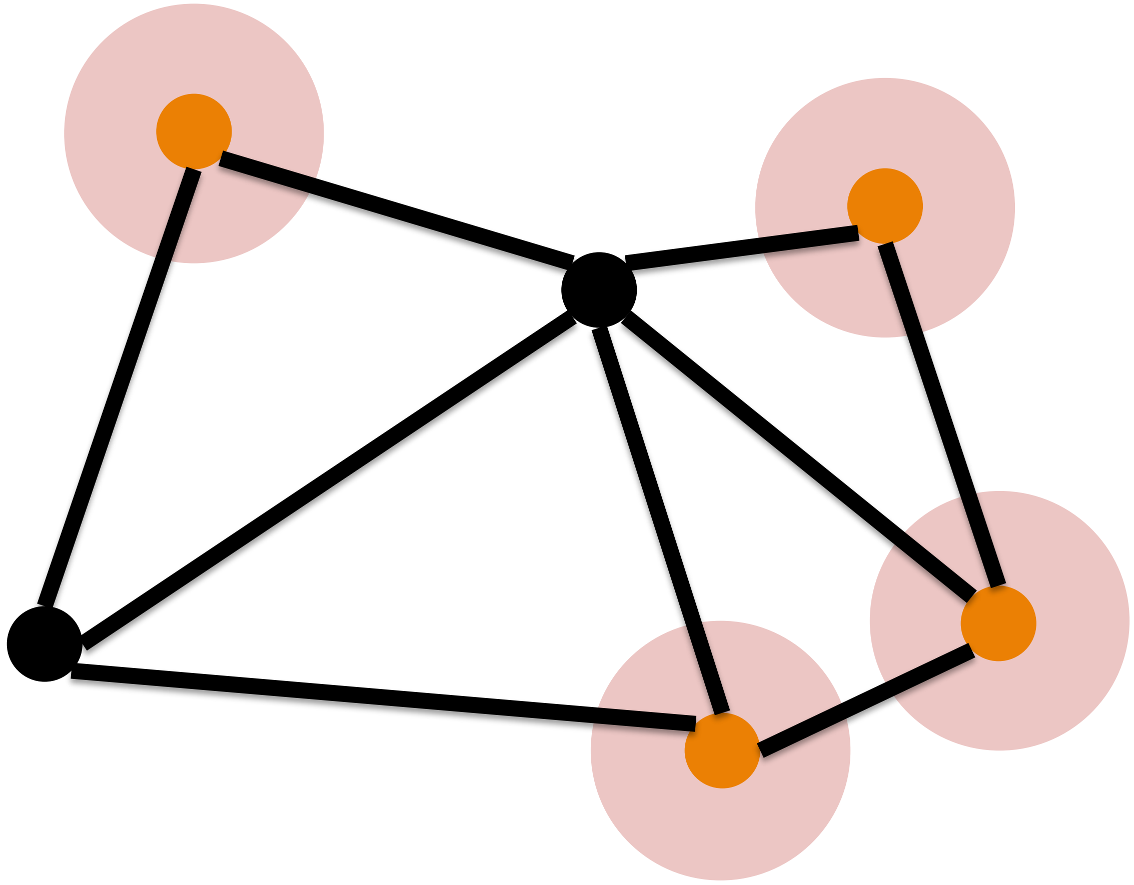
Rough algorithmic idea

- Increase dual variables of connected components
- When duals of two comps high enough to pay for a path to connect them, do so
- Can see this as process over time

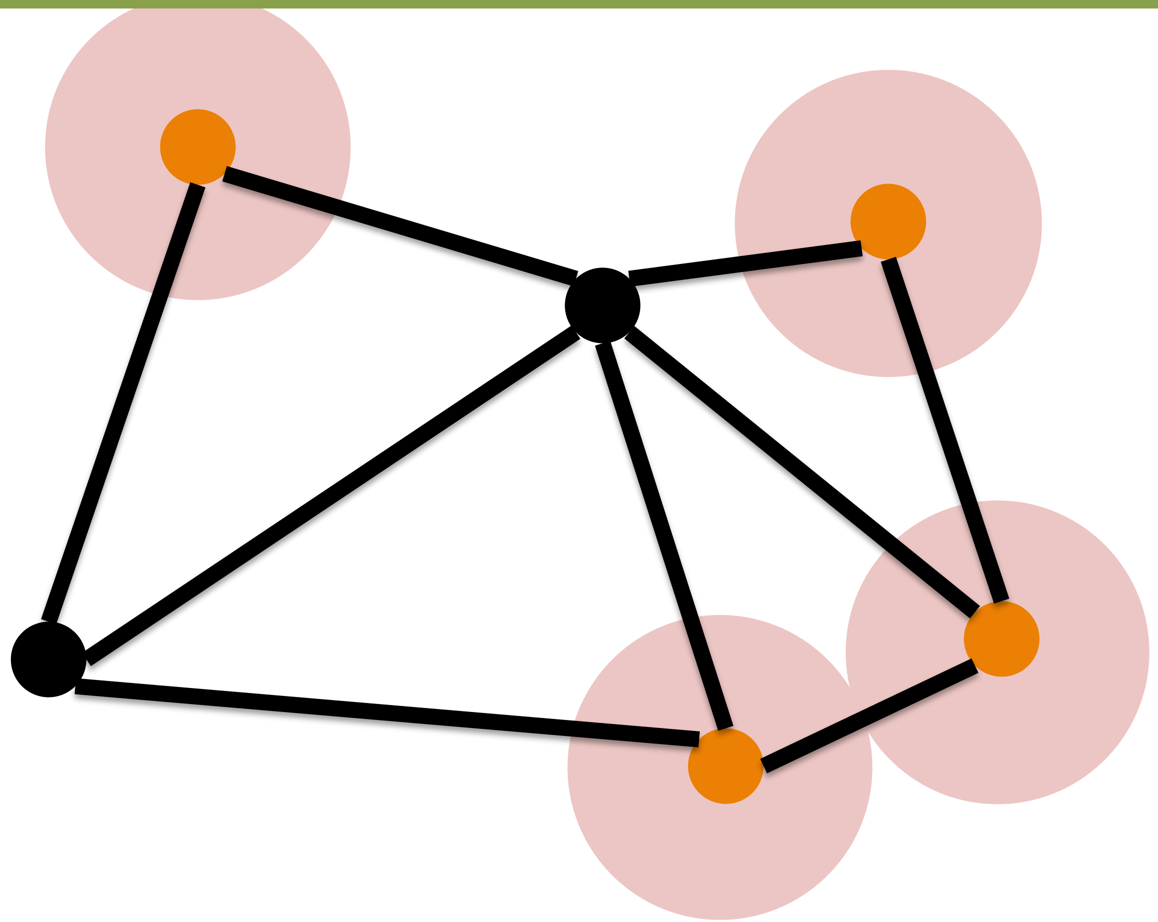
Example



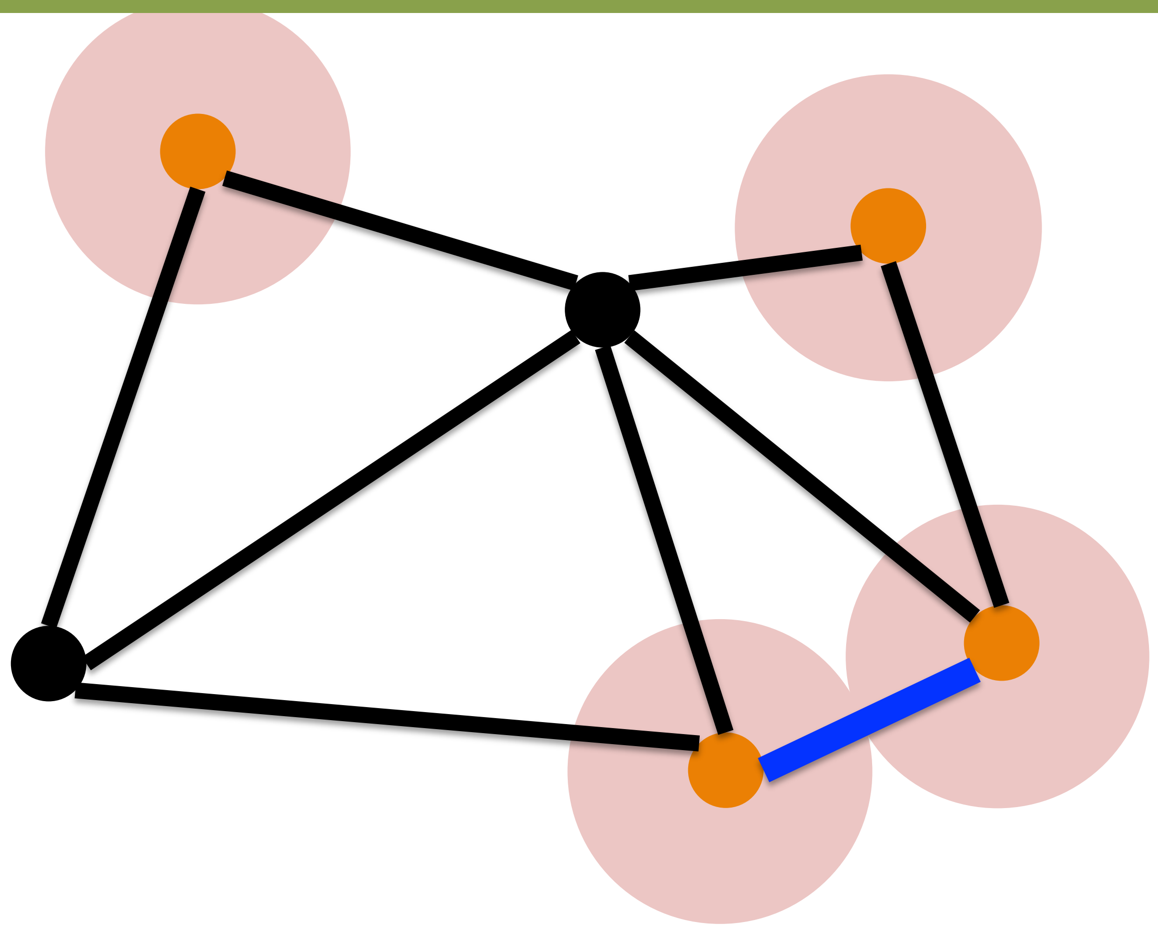
Example



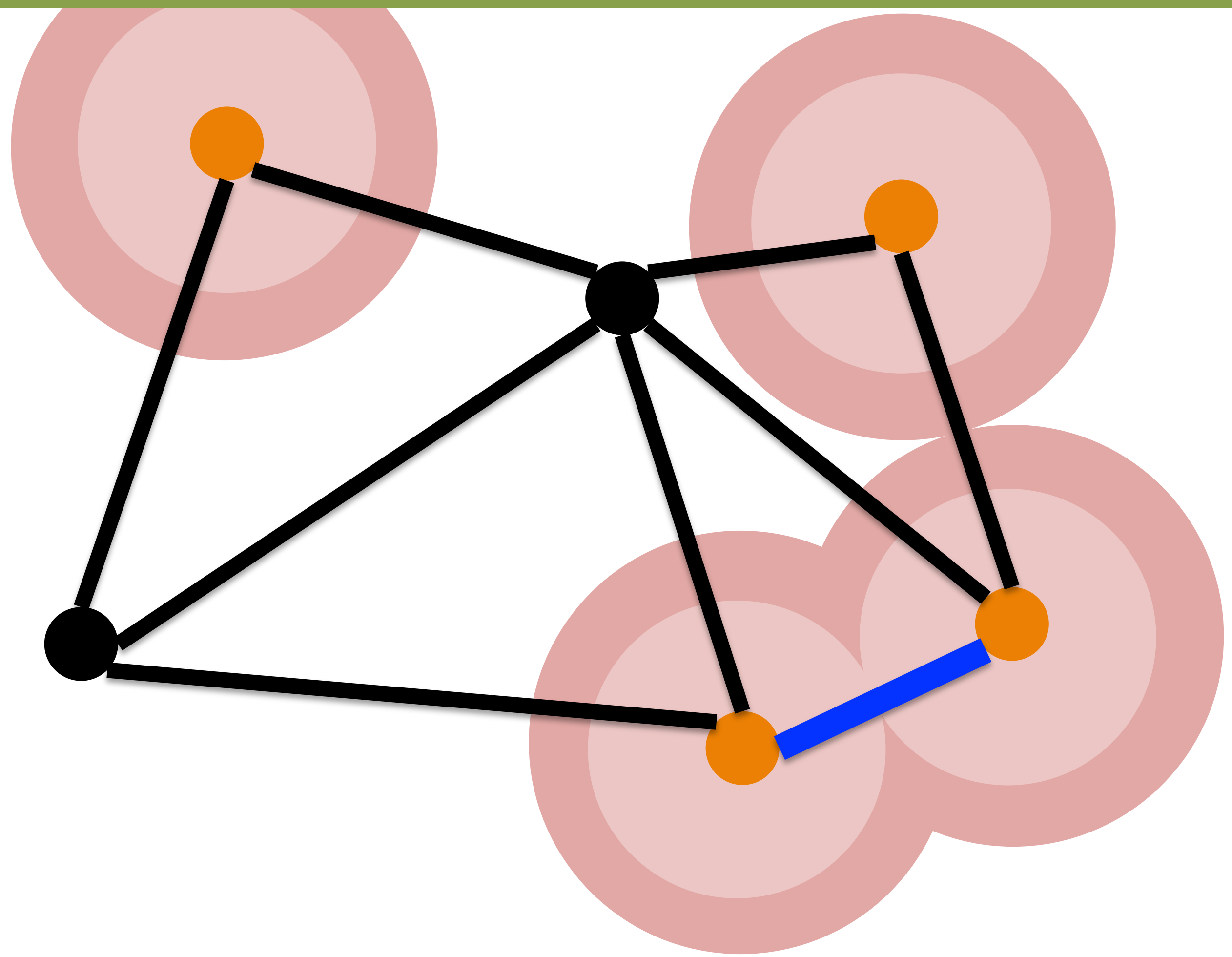
Example



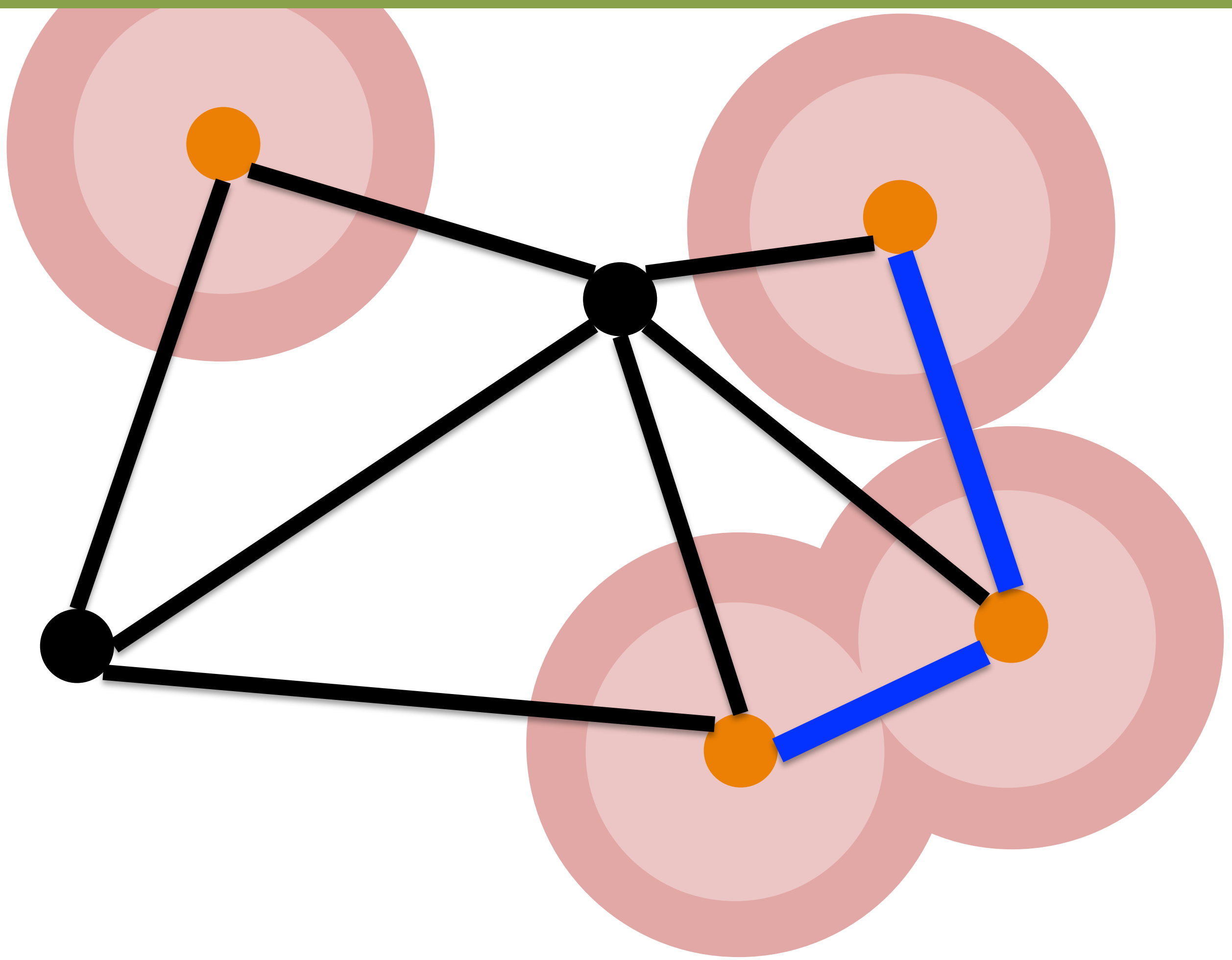
Example



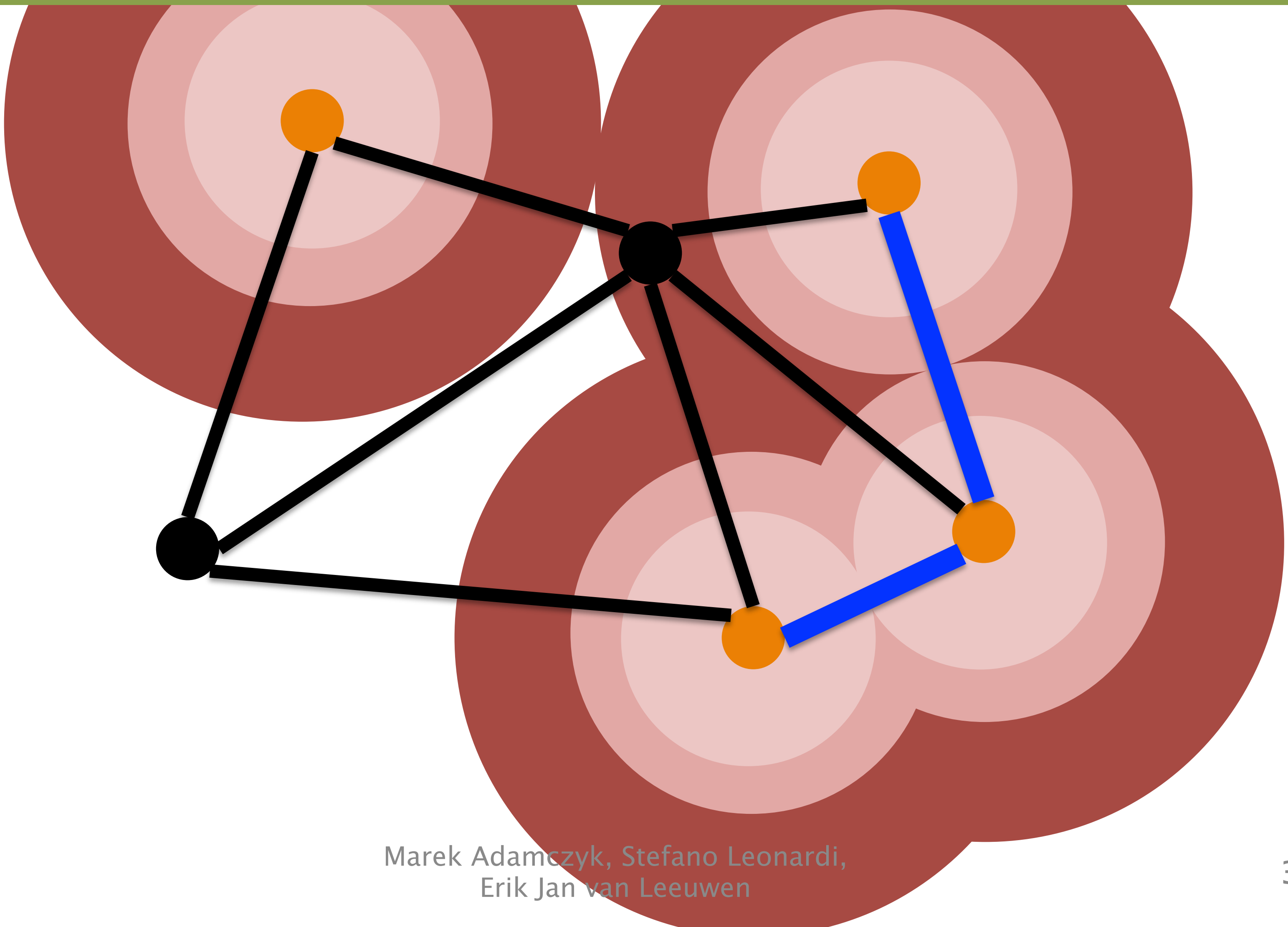
Example



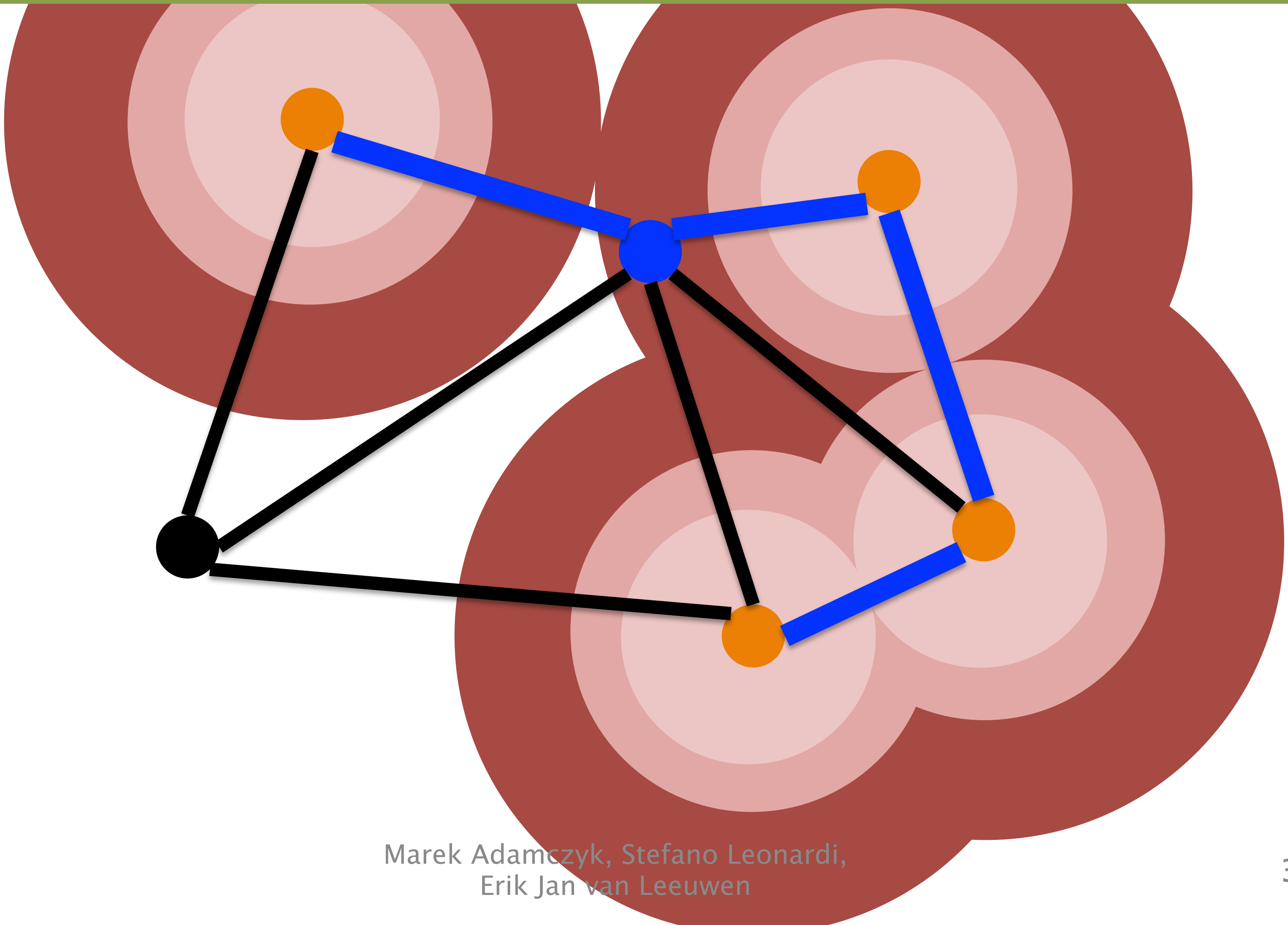
Example



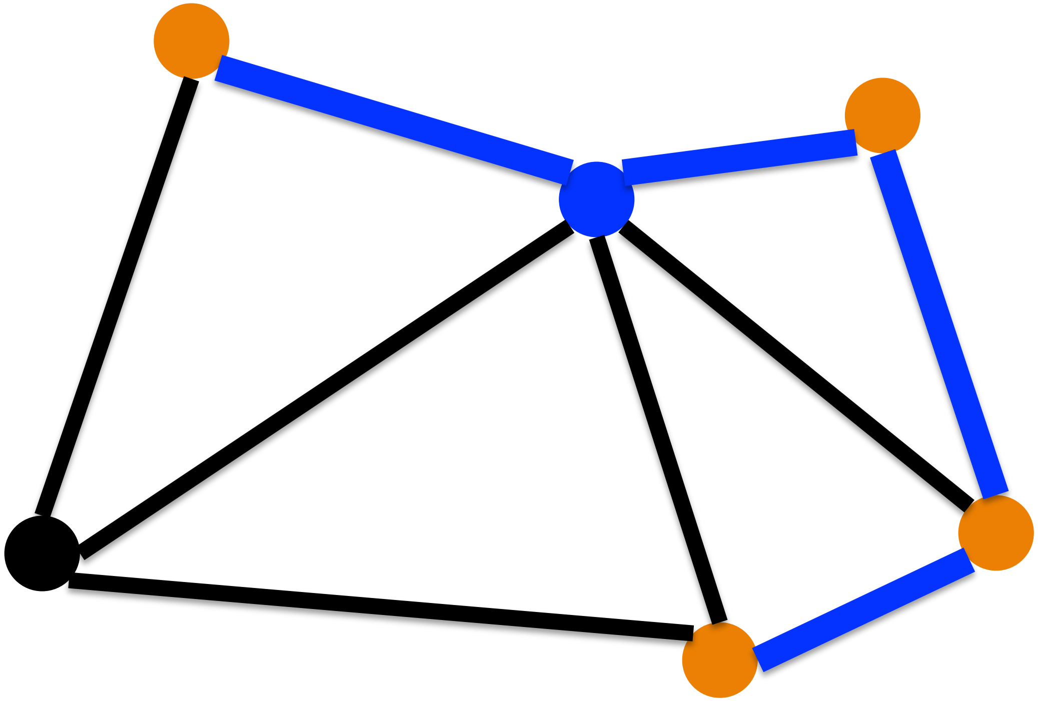
Example



Example



Example



Steiner Tree: primal-dual

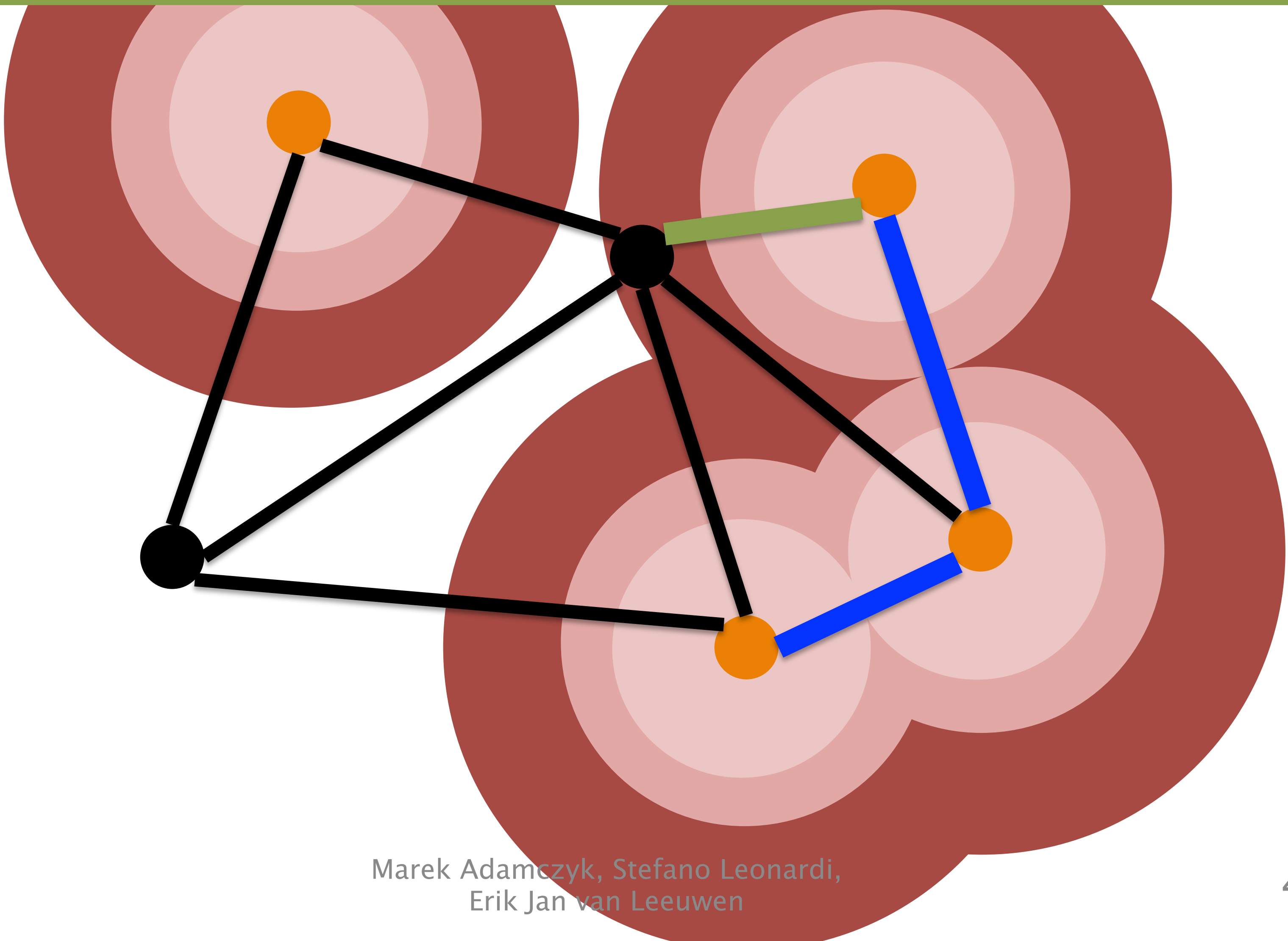
- $T=(R,\emptyset)$: tree being built
  $M = \{\{r\} \mid r \text{ in } R\}$: **active** comps.
  $y_S = 0$, $x_e = 0$, $t = 0$

- Simultaneously increase y_S for all C in M and t until $\Sigma_{S:e \text{ in } \delta(S)}\ y_S = c_e$ for some e (**tight** edge)

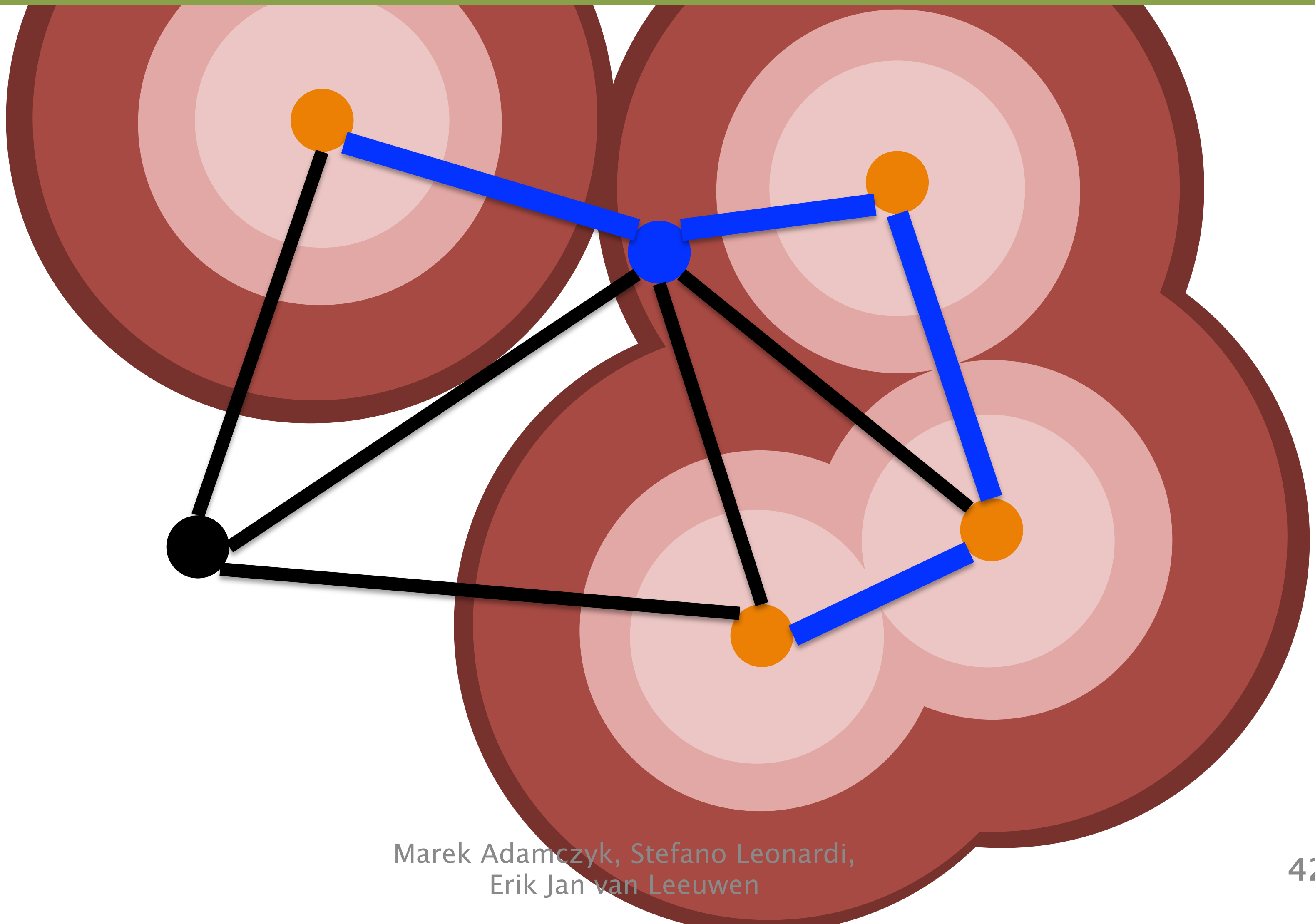
Tight edges

- $\Sigma_{S:e \text{ in } \delta(S)} y_S = c_e$; e in δ(C), C in M
- If e also in δ(C') for C' in M, then add C+C'+e to M, remove C,C'; C+C'+e contains r–r' path P for some r,r' unconnected in T, add P to T and set x_e=1 for all e in P
- Else, add C+e to M, remove C

Example



Example



Analysis

- Consider T_t and M_t: C and M at time t
- Let U be component of T_t and C be component of M_t containing U
- Claim: $c(U) \leq (\Sigma_{S \text{ subset } C}\ 2y_S) - 2t$ if U created at time t
  - Intuition: dual pays for roughly half of the primal solution

Analysis

- Claim: $c(U) \leq (\Sigma_{S \text{ subset } C}\ 2y_S) - 2t$
- Tight e between C_1 and C_2 of M_t; then $U=U_1+U_2+P$ for path P
- Claim: $c(P) \leq 2t$
- Cannot use other components of M to 'jump ahead'

Analysis

- Claim: $c(U) \leq (\Sigma_{S \text{ subset } C}\ 2y_S) - 2t$
- Tight e between C_1 and C_2 of M_t; then $U=U_1+U_2+P$ for path P
- $c(U) \leq c(U_1) + c(U_2) + 2t$
  $\leq(\Sigma_{S \text{ subset } C1,C2}\ 2y_S)-2t_1-2t_2+2t$
  $=(\Sigma_{S \text{ subset } C}\ 2y_S)-2(t-t_1)-2(t-t_2)-2t_1-2t_2+2t$
  $=(\Sigma_{S \text{ subset } C}\ 2y_S) - 2t$

Analysis

- Claim: $c(U) \leq (\Sigma_{S \text{ subset } C} 2y_S) - 2t$
- y is always feasible dual solution
  - After edge is tight: no y_S with e in $\delta(S)$ is raised again
- $\Sigma y_S \leq OPT \rightarrow 2 \Sigma y_S \leq 2\ OPT$
- $c(T) \leq (\Sigma_{S \text{ subset } C} 2y_S) - 2t \leq 2\ OPT$

Analysis

- T feasible, $c(T) \leq 2$ OPT
  - Better analysis: $c(T) \leq (2-1/|R|)$ OPT
- Polynomial time

Observations

- Lower bound shows why analysis is tight; even implies integrality gap is factor 2
- For R = V(G), primal-dual is Kruskal's algorithm for minimum spanning tree

More of primal-dual

Metric Facility Location

Technique: Primal-Dual

- Maintain feasible solution to dual
- Increase certain dual variables
- After ‘enough’ increase of dual, increase certain primal variables
- Continue until primal feasible

Metric Facility Location

- Graph (F,C,E): facilities F, clients C
- Facilities cost to open: f(i)
- Connection costs: c(e) **metric**, thus $c(uv) \leq c(ux) + c(xv)$
- Compute: set F' of open facilities and set E' of connections clients -> open facilities with f(F') + c(E') minimized

LP formulation

$$
\begin{array}{lll}
\text{minimize} & \sum_{i \in F} f(i) \cdot y_i + \sum_{i \in F, j \in C} c(ij) \cdot x_{ij} & \\
\text{such that} & \sum_{i \in F} x_{ij} \geq 1 & \forall j \in C \\
 & y_i - x_{ij} \geq 0 & \forall i \in F, j \in C \\
 & x_{ij} \in \{0, 1\} & \forall i \in F, j \in C \\
 & y_i \in \{0, 1\} & \forall i \in F
\end{array}
$$

x_{ij}: connect client j to facility i or not

y_i: open facility I or not

LP relaxation

$$\begin{array}{lll} \text{minimize} & \sum_{i \in F} f(i) \cdot y_i + \sum_{i \in F, j \in C} c(ij) \cdot x_{ij} & \\ \text{such that} & \sum_{i \in F} x_{ij} \geq 1 & \forall j \in C \\ & y_i - x_{ij} \geq 0 & \forall i \in F, j \in C \\ & x_{ij} \geq 0 & \forall i \in F, j \in C \\ & y_i \geq 0 & \forall i \in F \end{array}$$

LP dual

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{j \in C} \alpha_j & \\
\text{such that} & \alpha_j - \beta_{ij} \leq c_{ij} & \forall i \in F, j \in C \\
& \displaystyle\sum_{j \in C} \beta_{ij} \leq f_i & \forall i \in F \\
& \beta_{ij} \geq 0 & \forall i \in F, j \in C \\
& \alpha_j \geq 0 & \forall j \in C
\end{array}
$$

MFL: primal-dual idea

- Idea: start increasing α_j for all j (similar as before)
- $\alpha_j = c(ij)$: (ij **tight**) start increasing β_{ij} as well to remain dual feasible
- $\Sigma_{j \text{ in } C}\ \beta_{ij} = f(i)$: open facility i (it has been paid for)
- Consider process over time again

Algorithm: Phase 1

- Time 0, $\alpha_j = 0$, $\beta_{ij} = 0$
- Increase α_j of all unconn. clients
- If $\alpha_j = c(ij)$ and i open, conn. j to i
- If $\alpha_j = c(ij)$ and i closed, start increasing β_{ij} while α_j is increased
- $\Sigma_{j \text{ in } C}\, \beta_{ij} = f(i)$: open facility i and to i connect all unconn. j' with $\alpha_{j'} \geq c_{ij'}$

Algorithm: Phase 2

- For j, maybe $\beta_{ij} > 0$ for many i; but j cannot help pay for all of them
- Idea: open less facilities
- Facility i opened by algorithm is **permanently opened** if no other permanently opened facility i' satisfies $c(ij), c(i'j) < \alpha_j$ for some j (build maximal set)

Algorithm: output

- F' = permanently opened facilities
- E' = connect j to p.o. facility i in F' with $\alpha_j \geq c(ij)$ if possible (j is **good**); closest facility in F' otherwise (j is **bad**)

Observations

- No client j has $\beta_{ij} > 0$ and $\beta_{i'j} > 0$ for p.o. i, i'
  - Then $\alpha_j > c(ij)$ and $\alpha_j > c(i'j)$, contradicting i' and i are both p.o.
- Hence $\Sigma_{i \text{ in } F'} \Sigma_j \beta_{ij} = \Sigma_{i \text{ in } F'} f(i)$

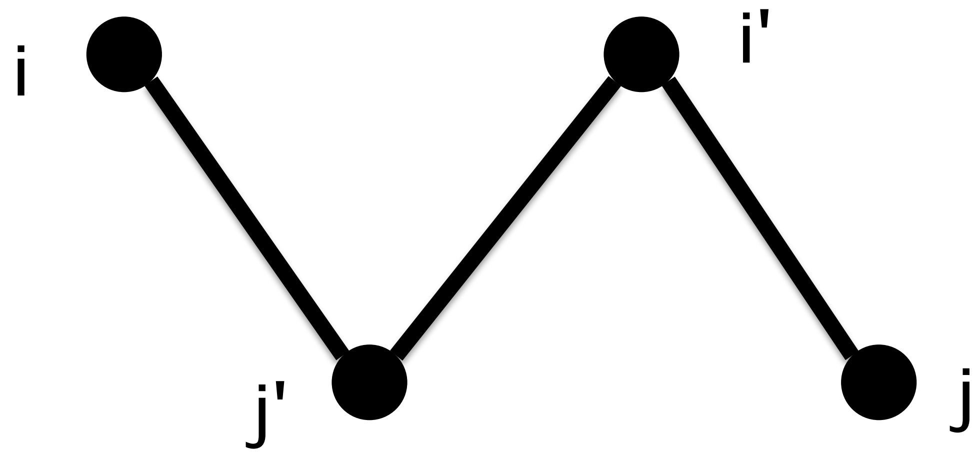
Observations

- Consider **good j**
- If j has $\beta_{ij} > 0$ for p.o.f. i, then j is good
- $\Sigma_{good\ j}\ \alpha_j = \Sigma_{good\ j}\ c(ij \text{ in } E') + \Sigma_{goodj}\ \beta_{ij \text{ in } E'}$ $= \Sigma_{good\ j}\ c(ij \text{ in } E') + \Sigma_{i \text{ in } F'}\ f(i)$

Observations

- Consider **bad j**
- Suppose j connected to i’ in algorithm. Why i’ not p.o.?

Observations



- i is p.o., $c(ij'), c(i'j') < \alpha_{j'}$; and $c(i'j) \leq \alpha_j$
- i opened at time t_i and i' opened at $t_{i'}$
- $\alpha_{j'} \leq \min\{t_i, t_{i'}\} \leq t_{i'} = \alpha_j$; thus $c(ij) \leq 3\ \alpha_j$

Analysis

- $\Sigma_{j \text{ in } C}$ c(ij in E') + $\Sigma_{i \text{ in } F'}$ f_i =
  $\Sigma_{\text{good } j}$ c(ij in E')+$\Sigma_{i \text{ in } F'}$ f_i+$\Sigma_{\text{bad } j}$ c(ij in E')
  = $\Sigma_{\text{good } j}$ α_j + 3 $\Sigma_{\text{bad } j}$ α_j
  = 3 $\Sigma_{j \text{ in } C}$ α_j
- So this gives factor 3 approximation in polynomial time

Running time

- Sort pairs ij by increasing cost; they go tight in this order
- Guess when facility is paid for and store these numbers in heap
- This gives order of events in algorithm

Running time

- Pair ij goes tight:
  1. i is open: j does not pay for other i'; update heap for all i' with new guess for when i' will be paid for
  2. i is closed: j starts to pay for i; update heap for i with new guess
- Facility i is paid for: for all j connected to i, do case 1 above

Running time

- Pair ij considered at most twice: when ij tight & when j connected
- Using binary heap: O(m log m)
  - m = |C| * |F|