Primal-Dual Algorithms for Network Design

Stefano Leonardi

Sapienza Universitá di Roma

Theoretical Computer Science – Academic year 2008/2009

Non Metric Facility location

Input:

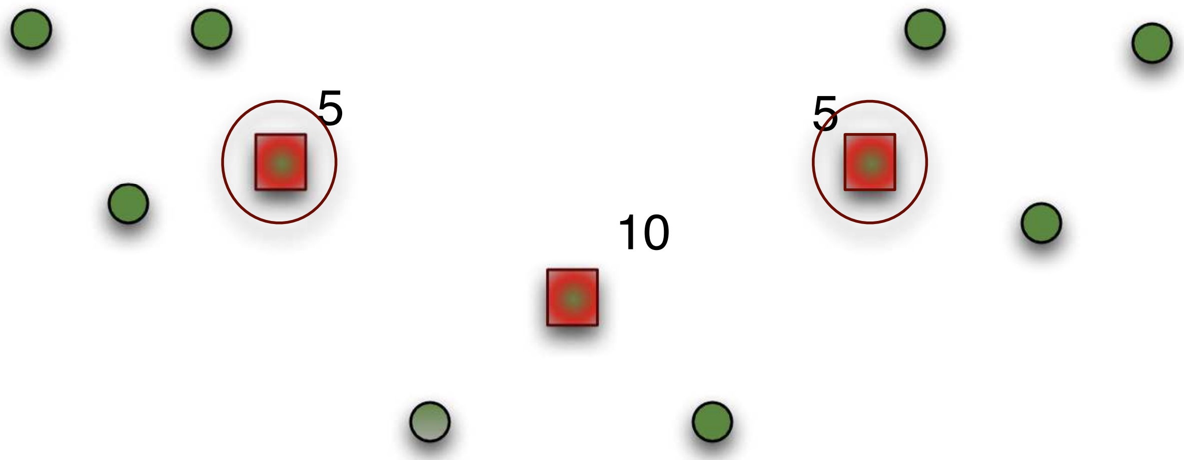
- undirected graph $G = (V, E)$
- non-negative edge costs $c : E \to \mathbb{R}^+$
- set of facilities $F \subseteq V$
- facility i has facility opening cost f_i
- set of demand points $D \subseteq V$
- c_{ij}: cost of connecting demand point j to facility i.
- Connection do not necessarily satisfy any metric: i.e. no triangle inequality: $c_{ij} \leq c_{ik} + c_{kj}$

Goal: Compute

- set $F' \subseteq F$ of opened facilities; and
- function $\phi : \mathcal{D} \to \mathcal{F}'$ assigning demand points to opened facilities that minimizes

$$\sum_{i \in F'} f_i + \sum_{j \in \mathcal{D}} c_{\phi(j)j}$$

Example



Two facilities of cost 5 are openend

Approximation on Non-metric Facility location

Algorithm 1: Algorithm for non-metric facility location.

Data: D, F, $c : D \times F \longrightarrow \Re_{\geq 0}$, $f : F \longrightarrow \Re_{\geq 0}$.

while $D \neq \emptyset$ **do**

  let $i \in F$ and $S \subseteq D$ minimize $\frac{f_i + \sum_{v \in S} c(v,i)}{|S \cap D|}$;

  $D \leftarrow D \setminus S$;

- Every step of the algorithm takes polynomial time since the most cost-effective facility is found between $|D| \times |F|$ different sets.
- Let S_i be the demand set that is covered at the ith iteration of the algorithm, $i = 1, \ldots, k$.
- Let $|C_i|$ be the number of uncovered demands before set S_i is selected.
- Denote by $c(S_i) = f_i + \sum_{v \in S_i} c(v, i)$ be the cost of the algorithm at the ith iteration.

Approximation on Non-metric Facility location

Theorem: The Greedy algorithm for Non-metric Facility Location is $O(\log n)$ approximate, with $n = |D|$.

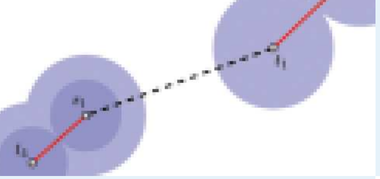
- The optimal solution will cover the demand set C_i at cost $\frac{c(OPT)}{|C_i|}$ per demand. Therefore there exists a set in the optimal solution of cost effectiveness lower than $\frac{c(OPT)}{|C_i|}$.
- The cost of the algorithm is bounded by

$$
\begin{aligned}
C(ALG) &\leq \sum_{i=1}^{k} cost(S_i) \leq c(OPT) \sum_{i=1}^{k} \frac{|S_i \cap C_i|}{|C_i|} \\
&\leq c(OPT) \sum_{i=1}^{|D|} \frac{1}{i} = O(\log n) c(OPT)
\end{aligned}
$$

- Primal-dual approximation algorithms construct a feasible dual together with an integral solution to the problem.

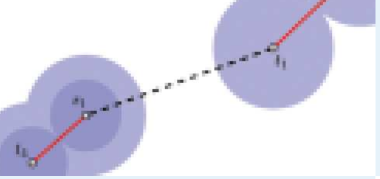
- Primal-dual approximation algorithms construct a feasible dual together with an integral solution to the problem.
- Approximation guarantee obtained by relating the cost of the integral solution to a feasible dual.

- Primal-dual approximation algorithms construct a feasible dual together with an integral solution to the problem.
- Approximation guarantee obtained by relating the cost of the integral solution to a feasible dual.



Primal-Dual Approximation Algorithms forNetwo

- Primal-dual approximation algorithms construct a feasible dual together with an integral solution to the problem.
- Approximation guarantee obtained by relating the cost of the integral solution to a feasible dual.



Primal-Dual Approximation Algorithms forNetwo

- Primal-dual approximation algorithms construct a feasible dual together with an integral solution to the problem.
- Approximation guarantee obtained by relating the cost of the integral solution to a feasible dual.

Metric Facility location

Input:

- undirected graph $G = (V, E)$
- non-negative edge costs $c : E \to \mathbb{R}^+$
- set of **facilities** $F \subseteq V$
- facility i has facility opening cost f_i
- set of **demand points** $D \subseteq V$
- c_{ij}: cost of connecting demand point j to facility i.
  Connection cost satisfy triangle inequality: $c_{ij} \leq c_{ik} + c_{kj}$

Goal: Compute

- set $F' \subseteq F$ of opened facilities; and
- function $\phi : \mathcal{D} \to \mathcal{F}'$ assigning demand points to opened facilities that minimizes

$$\sum_{i \in F'} f_i + \sum_{j \in \mathcal{D}} c_{\phi(j)j}$$

LP formulation

$$
\begin{array}{llcll}
\text{min} & \displaystyle\sum_{i\in F, j\in D} c_{ij}x_{ij} + \sum_{i\in F} f_i y_i & & & \\
\text{s.t.} & \displaystyle\sum_{i\in F} x_{ij} & \geq & 1 & j \in D \\
 & y_i - x_{ij} & \geq & 0 & i \in F, j \in D \\
 & x_{ij} & \in & \{0,1\} & i \in F, j \in D \\
 & y_i & \in & \{0,1\} & i \in F
\end{array}
$$

- $y_i = 1$ if facility i is opened;
- $x_{ij} = 1$ if demand j connected to facility i.

LP relaxation:

$$\text{min} \quad \sum_{i\in F, j\in D} c_{ij}x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i$$

$$\begin{array}{rrcll}
\text{s.t.} & \sum_{i\in F} x_{ij} & \geq & 1 & j \in D \\
& y_i - x_{ij} & \geq & 0 & i \in F, j \in D \\
& x_{ij} & \geq & 0 & i \in F, j \in D \\
& y_i & \geq & 0 & i \in F
\end{array}$$

$$\mathrm{DualProgram} : \text{max} \quad \sum_{j\in D} \alpha_j$$

$$\begin{array}{rrcll}
\text{s.t.} & \alpha_j - \beta_{ij} & \leq & c_{ij} & i \in F, j \in D \\
& \sum_{j\in D} \beta_{ij} & \leq & f_i & i \in F \\
& \alpha_j & \geq & 0 & j \in D \\
& \beta_{ij} & \geq & 0 & i \in F, j \in D
\end{array}$$

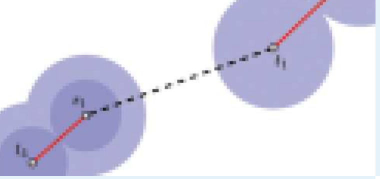
A 3-approximation algorithm

At time 0, set all $\alpha_j = 0$ and $\beta_{ij} = 0$ and declare all demands unconnected.

While there is an unconnected demand:

- Raise uniformly all α_j's of unconnected demands
- If $\alpha_j = c_{ij}$, declare demand j tight with facility i
- For a tight constraint ij, raise both α_j and β_{ij}
- If $\sum_j \beta_{ij} = f_i$ at time t_i, declare:
  - Facility i *temporarily opened* at time t_i;
  - All unconnected demands j that are tight with i *connected*;

[Jain and Vazirani, 1999][Mettu and Plaxton, 2000]



A 3-approximation algorithm

Opening facilities:

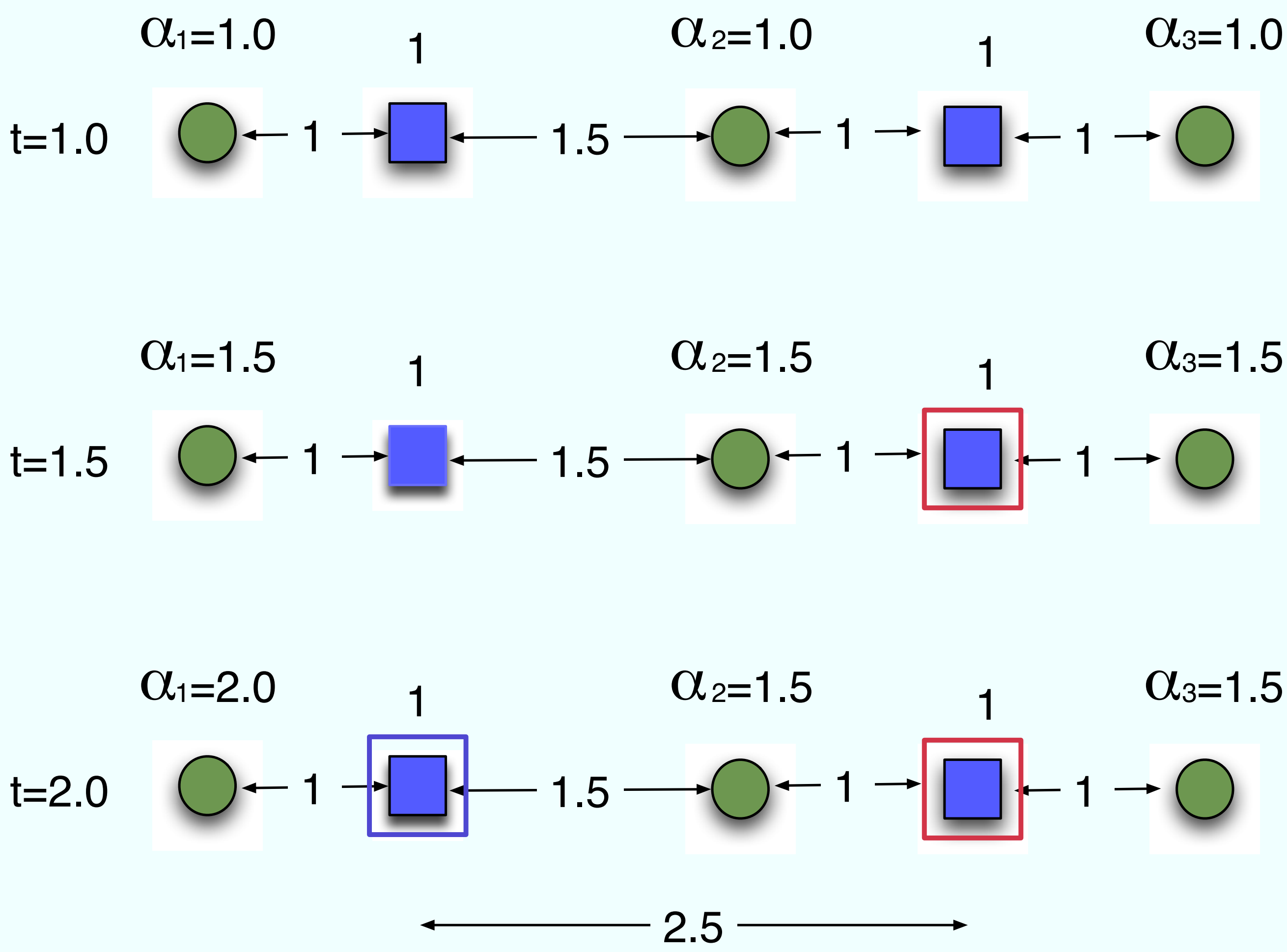
Demand points contribute to more permanently opened facilities. Not enough money for all of them.

- Facility i *temporarily opened* at time t_i;
- Declare facility i *permanently opened* if there is no permanently opened facility within distance $2t_i$.

Open all permanently opened facilities.

Connect each demand to the nearest opened facility.

Example of execution of the algorithm



Proof of 3 approximation.

Demands connected to opened facilities

- $\alpha_j = c_{ij} + \beta_{ij}$ for demands connected to opened facility i.
- α_j pays for connection cost c_{ij} and contribute with β_{ij} to f_i.
- Since other opened facilities are at distance $> t_i$, α_j does not pay for opening any other facility.

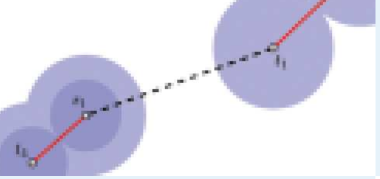
Demands connected to temporarily opened facilities

- Demand j connected to temporarily opened facility i. There exists an opened facility i' with $c_{ii'} \leq 2t_i$.
- Since $c_{ji} \leq \alpha_j$ and $t_i \leq \alpha_j$, $c_{ji'} \leq c_{ji} + c_{ii'} \leq 3\alpha_j$

Steiner trees

The Steiner tree problem has been defined for the first time by Gauss in a letter to Schumacher

- **S**teiner trees



Steiner trees

The Steiner tree problem has been defined for the first time by Gauss in a letter to Schumacher

- **S**teiner trees

  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \rightarrow \mathbb{R}^+$;
  - terminal-set $R = \{s_1, \ldots, s_k\} \subseteq V$.
  - Steiner vertices V/R

Steiner trees

The Steiner tree problem has been defined for the first time by Gauss in a letter to Schumacher

- **S**teiner trees

  Input:

  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \rightarrow \mathbb{R}^+$;
  - terminal-set $R = \{s_1, \ldots, s_k\} \subseteq V$.
  - Steiner vertices V/R

  Goal:

  Compute min-cost tree T in G that contains all vertices in R and any subset of the Steiner vertices.

Steiner trees

The Steiner tree problem has been defined for the first time by Gauss in a letter to Schumacher

- **S**teiner trees

  Input:

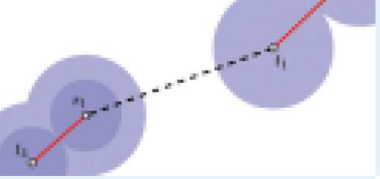
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \rightarrow \mathbb{R}^+$;
  - terminal-set $R = \{s_1, \ldots, s_k\} \subseteq V$.
  - Steiner vertices V/R

  Goal:

  Compute min-cost tree T in G that contains all vertices in R and any subset of the Steiner vertices.

Steiner trees

- We will consider the Steiner tree problem on metric spaces, i.e. $c(u, v) \leq c(u, w) + c(w, v)$.
- There exists a cost preserving reduction from Steiner tree to metric Steiner tree.
- Metric closure of G is the complete graph G' with costs $c'(u, v)$ equal to the shortest u, v path in G.
- We can transform in polynomial time an instance I of Steiner tree in G into an instance I' of Steiner tree in G' Prove!
- A solution of a given cost to instance I' in G' can be transformed into solution of no higher cost to instance I in G Prove!
- A ρ approximation to I' in G' can be transformed into a ρ approximation to I in G.



Steiner trees

- We will consider the Steiner tree problem on metric spaces, i.e. $c(u,v) \leq c(u,w) + c(w,v)$.
- There exists a cost preserving reduction from Steiner tree to metric Steiner tree.
- Metric closure of G is the complete graph G' with costs $c'(u,v)$ equal to the shortest u,v path in G.
- We can transform in polynomial time an instance I of Steiner tree in G into an instance I' of Steiner tree in G' Prove!
- A solution of a given cost to instance I' in G' can be transformed into solution of no higher cost to instance I in G Prove!
- A ρ approximation to I' in G' can be transformed into a ρ approximation to I in G.

Steiner trees

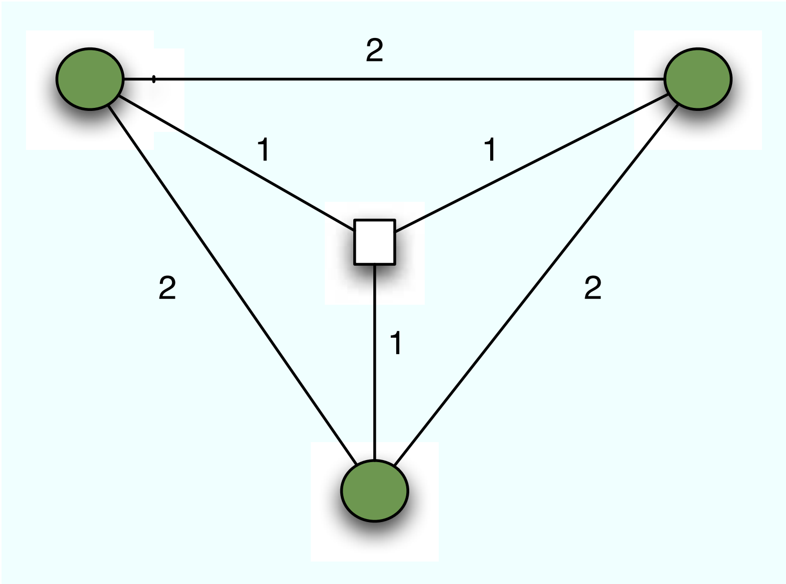
- We will consider the Steiner tree problem on metric spaces, i.e. $c(u, v) \leq c(u, w) + c(w, v)$.
- There exists a cost preserving reduction from Steiner tree to metric Steiner tree.
- Metric closure of G is the complete graph G' with costs $c'(u, v)$ equal to the shortest u, v path in G.
- We can transform in polynomial time an instance I of Steiner tree in G into an instance I' of Steiner tree in G' Prove!
- A solution of a given cost to instance I' in G' can be transformed into solution of no higher cost to instance I in G Prove!
- A ρ approximation to I' in G' can be transformed into a ρ approximation to I in G.

Steiner trees

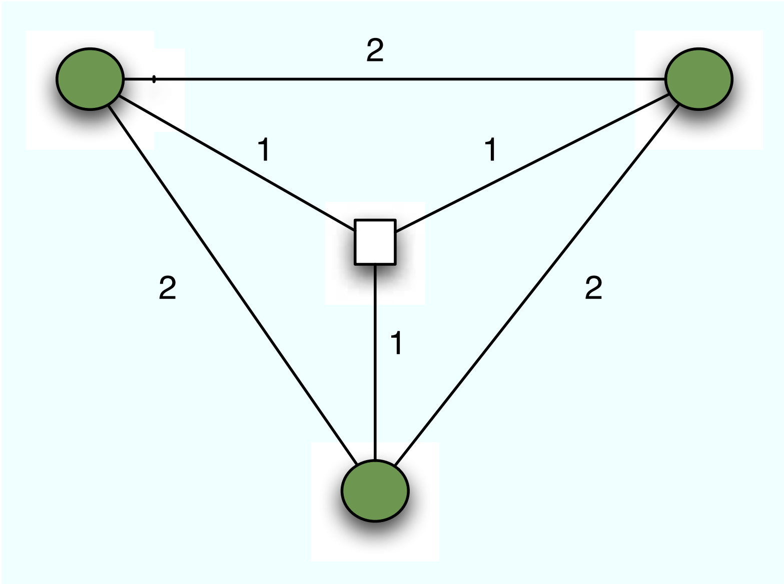
- We will consider the Steiner tree problem on metric spaces, i.e. $c(u,v) \leq c(u,w) + c(w,v)$.
- There exists a cost preserving reduction from Steiner tree to metric Steiner tree.
- Metric closure of G is the complete graph G' with costs $c'(u,v)$ equal to the shortest u,v path in G.
- We can transform in polynomial time an instance I of Steiner tree in G into an instance I' of Steiner tree in G' Prove!
- A solution of a given cost to instance I' in G' can be transformed into solution of no higher cost to instance I in G Prove!
- A ρ approximation to I' in G' can be transformed into a ρ approximation to I in G.

The MST heuristic for Steiner trees

- The MST of vertices R in G' returns a feasible solution of no larger cost for the Steiner tree problem on I in G
- The MST can in general be costlier than the Steiner tree. The MST problem is indeed solvable in polynomial time whereas Steiner tree is NP-hard.
- However, we can also relate the cost of the MST to the cost of the optimal Steiner tree

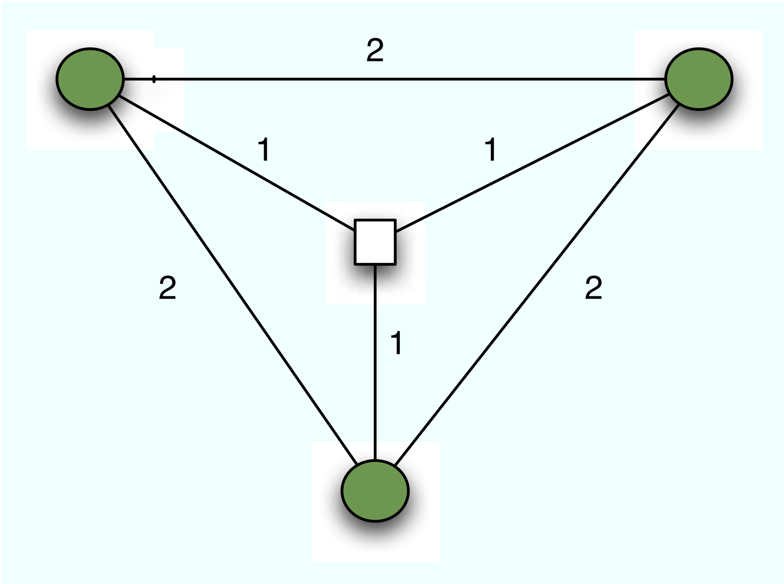


- The MST of vertices R in G' returns a feasible solution of no larger cost for the Steiner tree problem on I in G
- The MST can in general be costlier than the Steiner tree. The MST problem is indeed solvable in polynomial time whereas Steiner tree is NP-hard.
- However, we can also relate the cost of the MST to the cost of the optimal Steiner tree

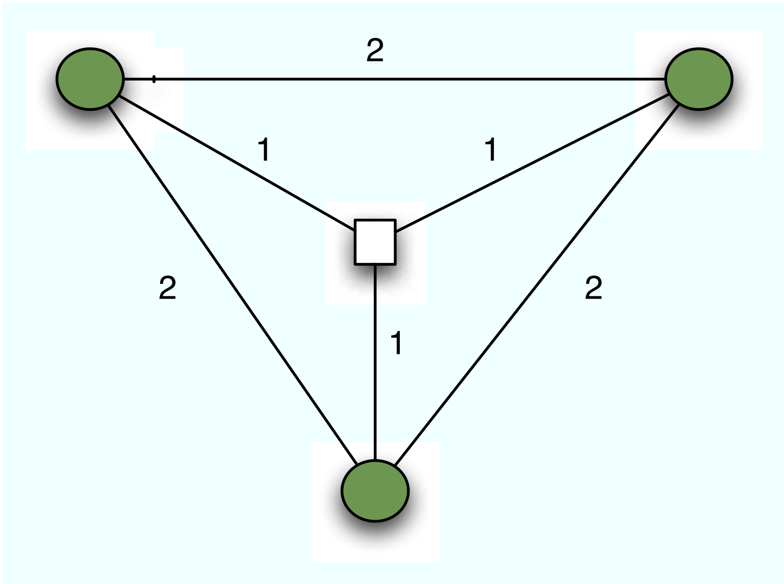


The MST heuristic for Steiner trees

- The MST of vertices R in G' returns a feasible solution of no larger cost for the Steiner tree problem on I in G
- The MST can in general be costlier than the Steiner tree. The MST problem is indeed solvable in polynomial time whereas Steiner tree is NP-hard.
- However, we can also relate the cost of the MST to the cost of the optimal Steiner tree



- The MST of vertices R in G' returns a feasible solution of no larger cost for the Steiner tree problem on I in G
- The MST can in general be costlier than the Steiner tree. The MST problem is indeed solvable in polynomial time whereas Steiner tree is NP-hard.
- However, we can also relate the cost of the MST to the cost of the optimal Steiner tree



The MST heuristic for Steiner trees

- **Theorem:** The cost of the MST on R in G' is at most twice the cost of the optimal Steiner tree of R in G.

  Proof:
- Consider for the analysis an optimal Steiner tree of R in G.
- Double all the edges to construct an Eulerian graph that connects all the vertices of R.
- Find an Eulerian tour with a DFS traversing of the edges of the Eulerian graph.
- Obtain a Hamiltonian cycle by shortcutting the Steiner vertices and the vertices of R already visited by the cycle. The short-cutting is done without increasing the cost of the eulerian tour given the triangle inequality.
- Obtain a Spanning tree by deleting one edge of the Hamiltonian cycle.
- **Claim:** There exists a Spanning tree of R on G' of equal cost
- Therefore the MST of R in G' is of cost at most $2 \times OPT$

The MST heuristic for Steiner trees

- **Theorem:** The cost of the MST on R in G' is at most twice the cost of the optimal Steiner tree of R in G.

  Proof:
- Consider for the analysis an optimal Steiner tree of R in G.
- Double all the edges to construct an Eulerian graph that connects all the vertices of R.
- Find an Eulerian tour with a DFS traversing of the edges of the Eulerian graph.
- Obtain a Hamiltonian cycle by shortcutting the Steiner vertices and the vertices of R already visited by the cycle. The short-cutting is done without increasing the cost of the eulerian tour given the triangle inequality.
- Obtain a Spanning tree by deleting one edge of the Hamiltonian cycle.
- **Claim:** There exists a Spanning tree of R on G' of equal cost
- Therefore the MST of R in G' is of cost at most $2 \times OPT$

The MST heuristic for Steiner trees

- **Theorem:** The cost of the MST on R in G' is at most twice the cost of the optimal Steiner tree of R in G.

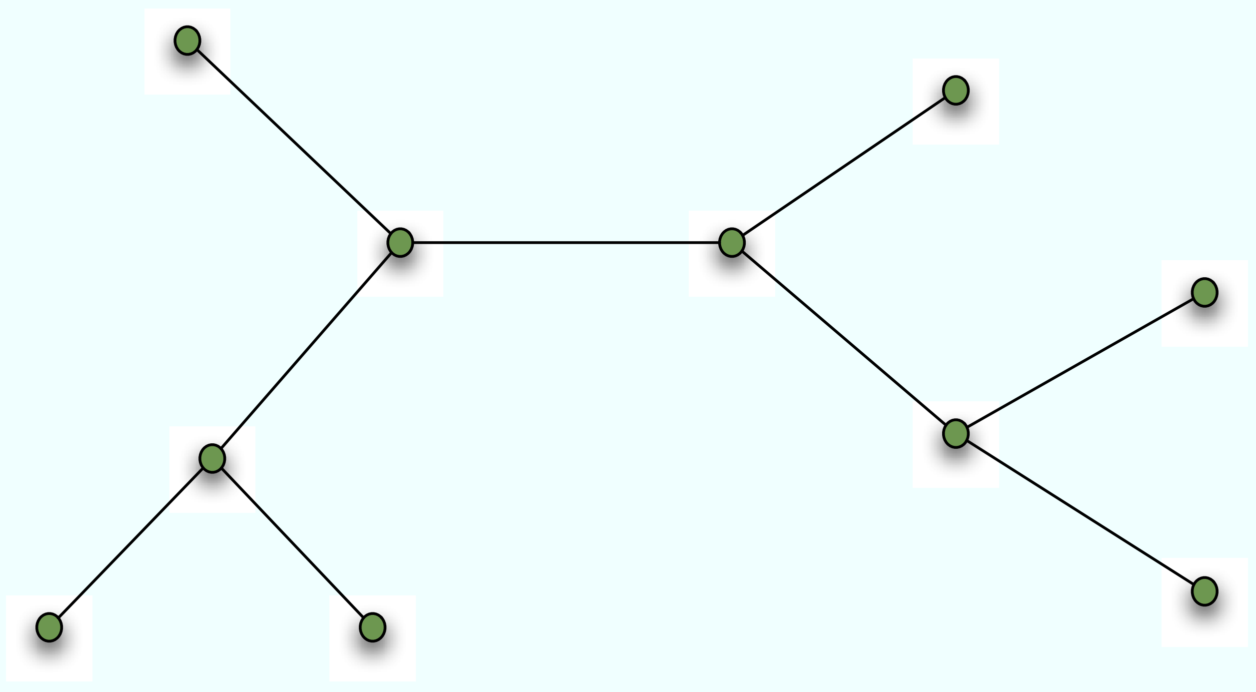
  Proof:
- Consider for the analysis an optimal Steiner tree of R in G.
- Double all the edges to construct an Eulerian graph that connects all the vertices of R.
- Find an Eulerian tour with a DFS traversing of the edges of the Eulerian graph.
- Obtain a Hamiltonian cycle by shortcutting the Steiner vertices and the vertices of R already visited by the cycle. The short-cutting is done without increasing the cost of the eulerian tour given the triangle inequality.
- Obtain a Spanning tree by deleting one edge of the Hamiltonian cycle.
- **Claim:** There exists a Spanning tree of R on G' of equal cost
- Therefore the MST of R in G' is of cost at most $2 \times OPT$

The MST heuristic for Steiner trees

- **Theorem:** The cost of the MST on R in G' is at most twice the cost of the optimal Steiner tree of R in G.

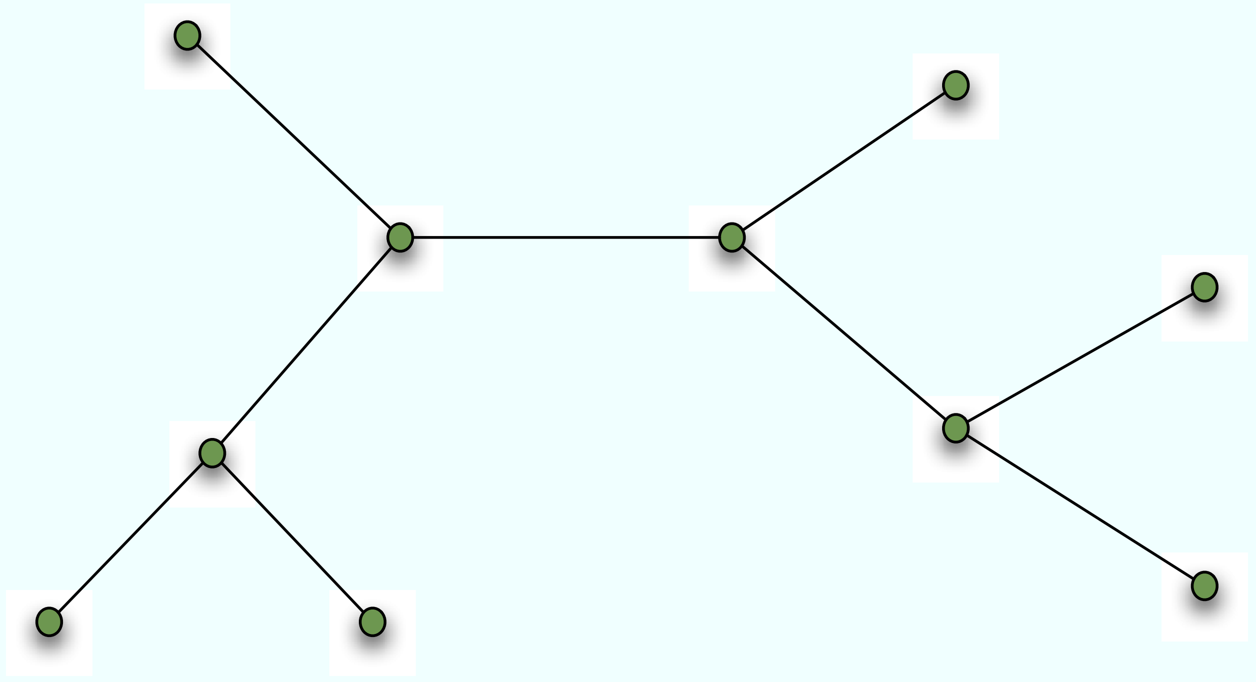
  Proof:
- Consider for the analysis an optimal Steiner tree of R in G.
- Double all the edges to construct an Eulerian graph that connects all the vertices of R.
- Find an Eulerian tour with a DFS traversing of the edges of the Eulerian graph.
- Obtain a Hamiltonian cycle by shortcutting the Steiner vertices and the vertices of R already visited by the cycle. The short-cutting is done without increasing the cost of the eulerian tour given the triangle inequality.
- Obtain a Spanning tree by deleting one edge of the Hamiltonian cycle.
- **Claim:** There exists a Spanning tree of R on G' of equal cost
- Therefore the MST of R in G' is of cost at most $2 \times OPT$

The MST heuristic for Steiner trees

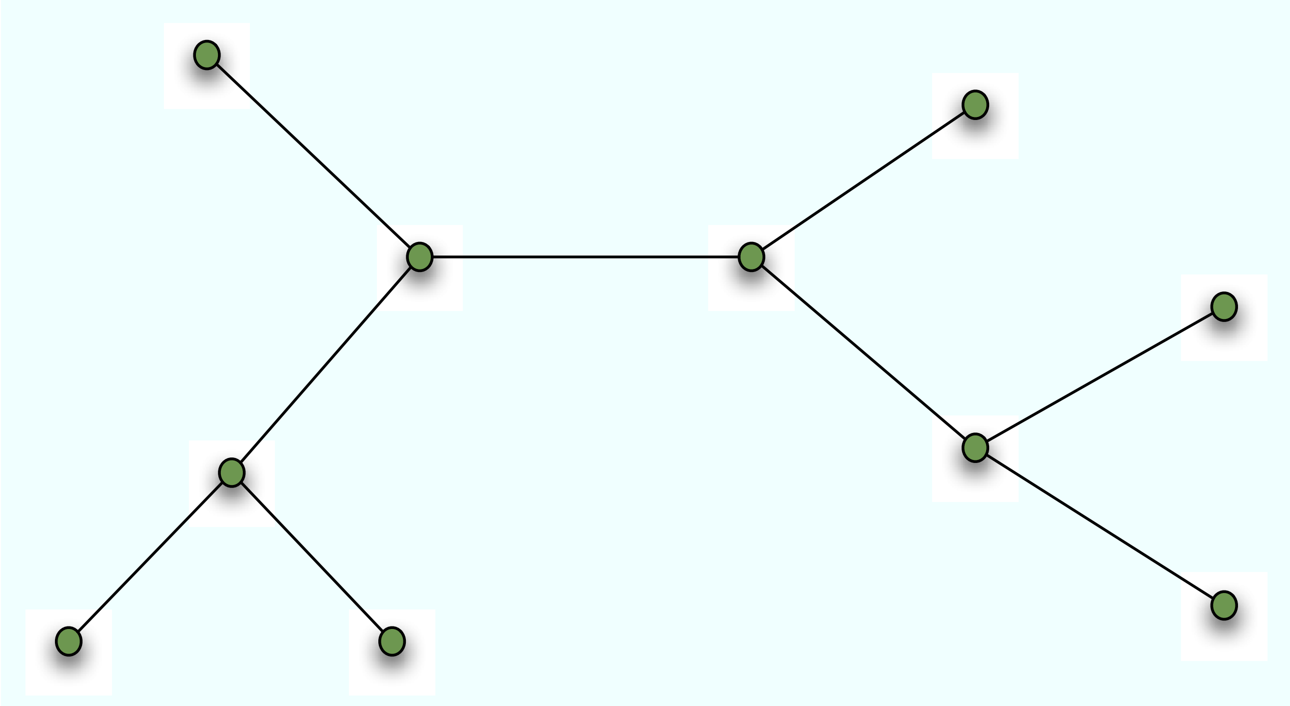


The Optimal Steiner tree

The MST heuristic for Steiner trees

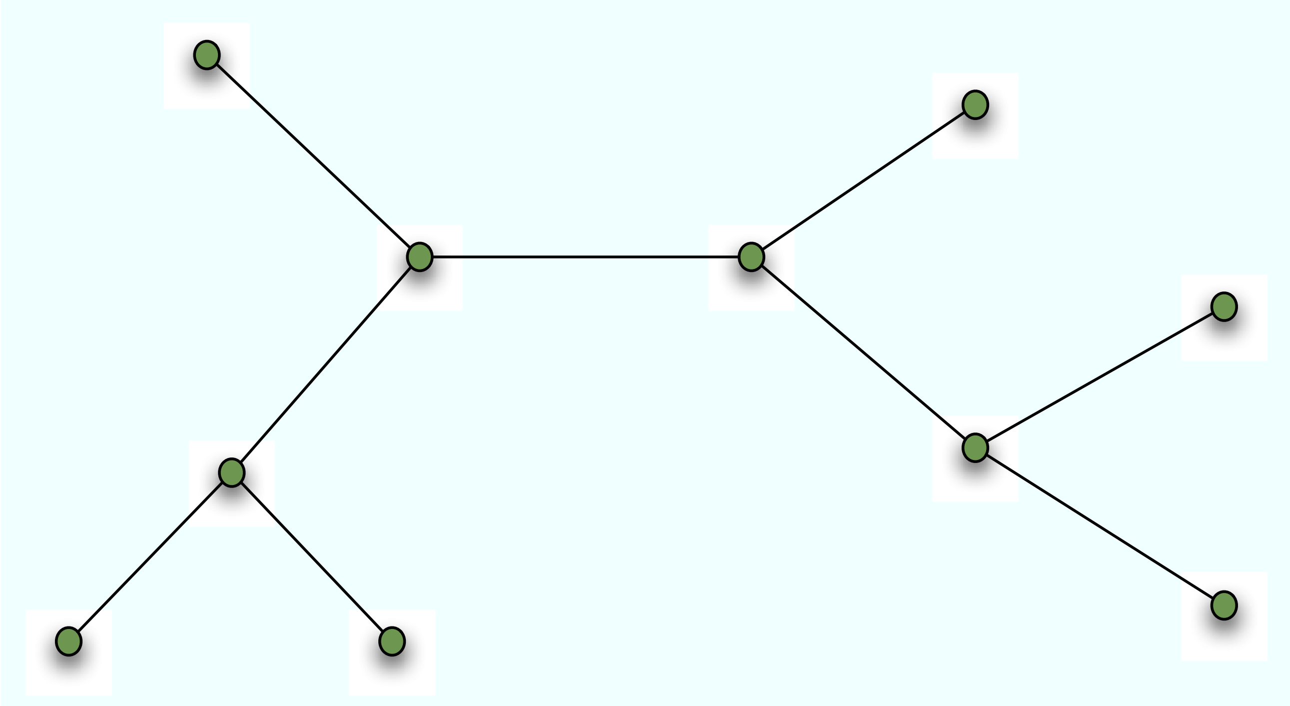


The Optimal Steiner tree



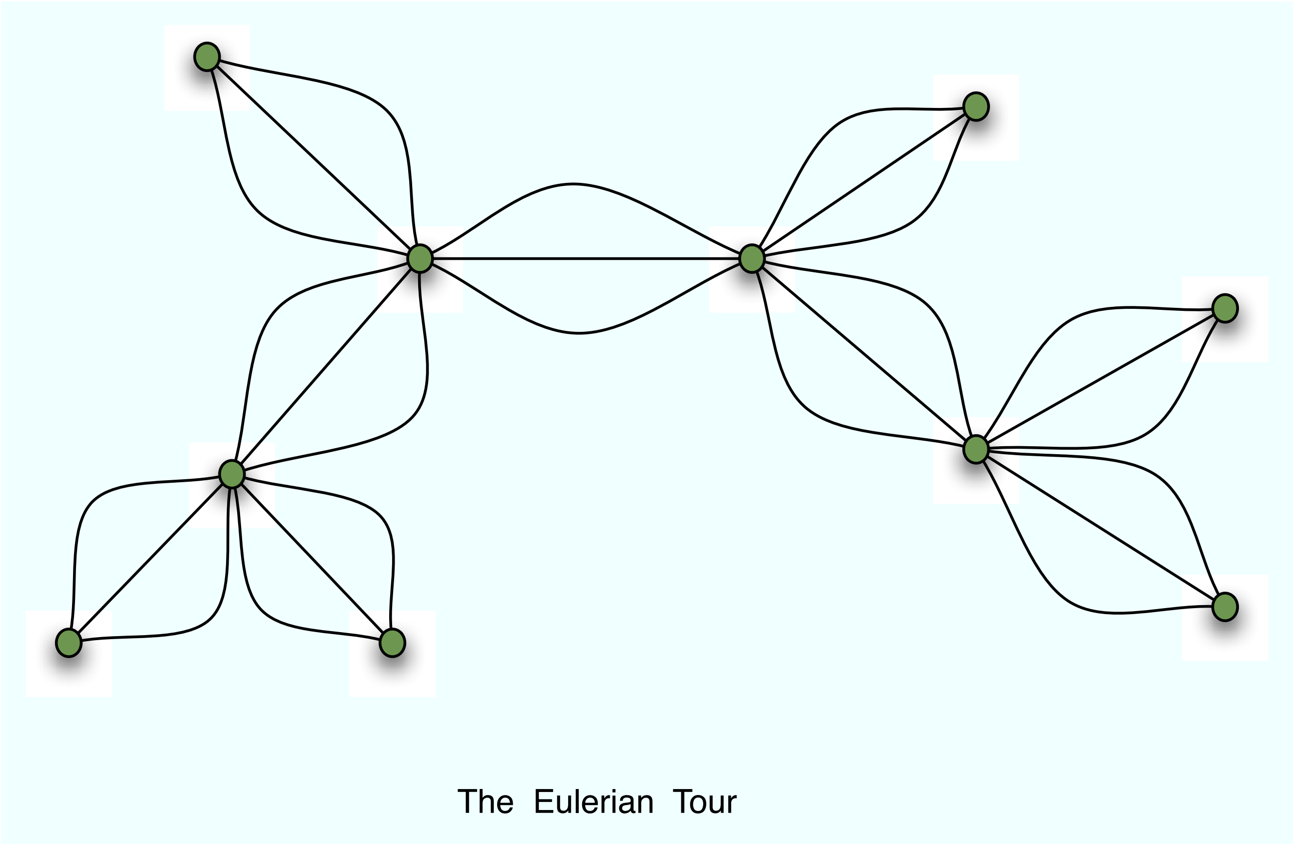
The Optimal Steiner tree

The MST heuristic for Steiner trees

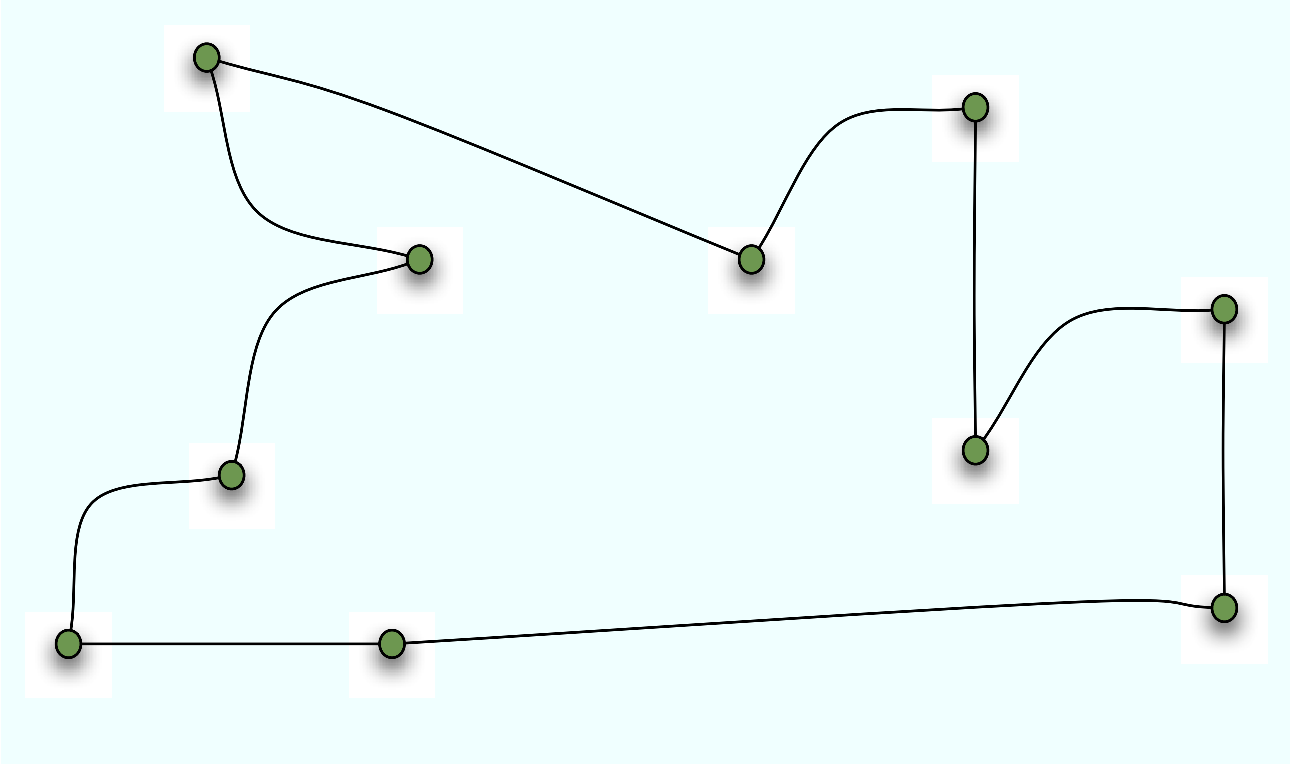


The Optimal Steiner tree

The MST heuristic for Steiner trees



The MST heuristic for Steiner trees



The Hamiltonian Cycle

Steiner Forests

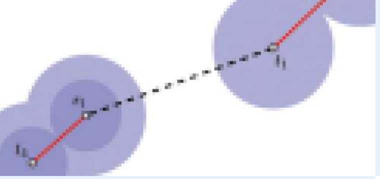
Steiner forests

- **S**teiner forests

Steiner forests

- **Steiner forests**

  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \to \mathbb{R}^+$;
  - terminal-pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$.



Steiner forests

- **S**teiner forests

  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \to \mathbb{R}^+$;
  - terminal-pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$.

  Goal:

  Compute min-cost forest F in G such that s and t are in same tree for all $(s, t) \in R$.

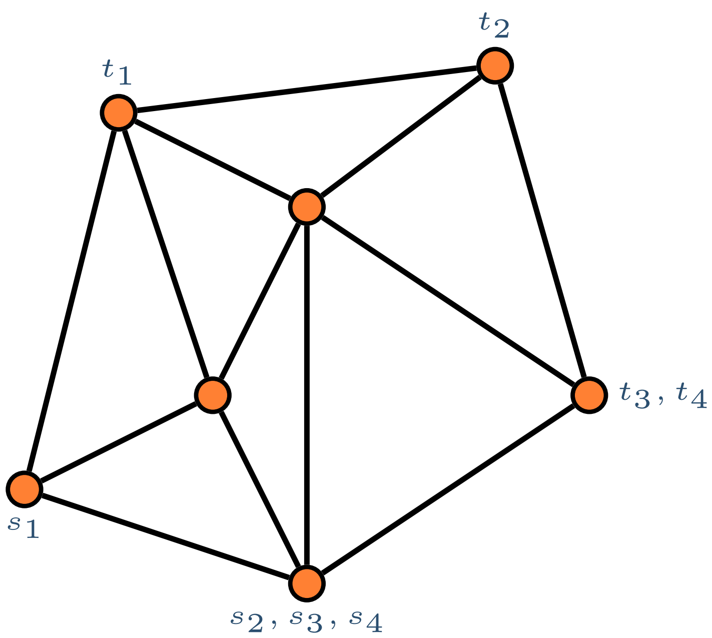
Steiner forests

- **S**teiner forests
  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \rightarrow \mathbb{R}^+$;
  - terminal-pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$.

  Goal:
  Compute min-cost forest F in G such that s and t are in same tree for all $(s, t) \in R$.
- Special case: Steiner trees.
  Compute a min-cost tree spanning a teminal-set $R \subseteq V$.

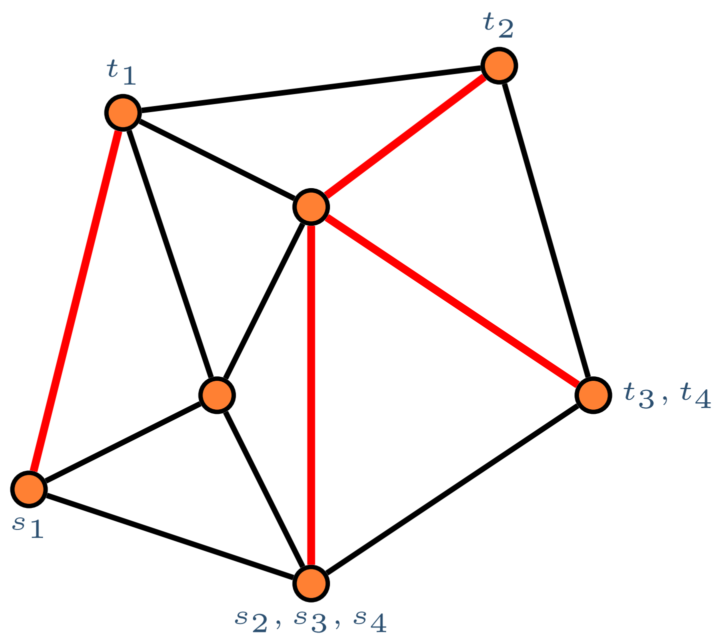
Steiner forests: Example

- Example with four terminal pairs: $R = \{(s_i, t_i)\}_{1 \le i \le 4}$
- All edges have unit cost.



Steiner forests: Example

- Example with four terminal pairs: $R = \{(s_i, t_i)\}_{1 \leq i \leq 4}$
- All edges have unit cost.



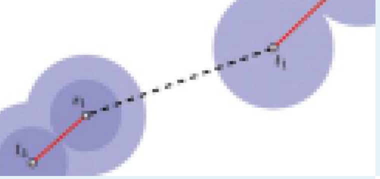
Total cost is 4!

Previous Work

- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])

Previous Work

- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])
- The Goemans and Williamson algorithm applies to a wider set of network design problem



Previous Work

- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])
- The Goemans and Williamson algorithm applies to a wider set of network design problem
- These are cornerstones Primal-dual methods

Previous Work

- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])
- The Goemans and Williamson algorithm applies to a wider set of network design problem
- These are cornerstones Primal-dual methods
- We’ll present the AKR algorithm and its analysis and then the GW algorithm and its analysis.

Steiner Forests: Primal-dual algorithm

- We sketch primal-dual algorithm SF due to [Agrawal, Klein, Ravi '95] (see also [Goemans, Williamson '95]).

Steiner Forests: Primal-dual algorithm

- We sketch primal-dual algorithm SF due to [Agrawal, Klein, Ravi '95] (see also [Goemans, Williamson '95]).
- Algorithm SF computes
  - feasible Steiner forest F, and
  - feasible dual solution y

  at the same time.

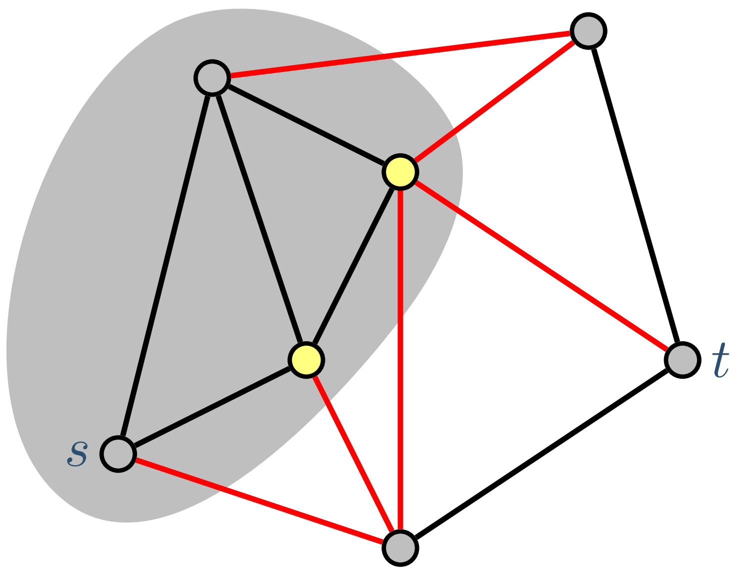
  Key trick: Use dual y and weak duality to bound cost of F.

Primal LP: Steiner Cuts

- Primal has variables x_e for all $e \in E$.
  $x_e = 1$ if e is in Steiner forest, 0 otherwise

Primal LP: Steiner Cuts

- Primal has variables x_e for all $e \in E$.
  $x_e = 1$ if e is in Steiner forest, 0 otherwise
- Steiner cut: Subset of nodes that separates at least one terminal pair $(s, t) \in R$.



Any feasible Steiner forest must contain at least one of the red edges!

Primal LP: Steiner Cuts

Primal LP has one constraint for each Steiner cut.

$$
\begin{array}{llcll}
\text{min} & \displaystyle\sum_{e \in E} c_e x_e & & & \\
\text{s.t.} & \displaystyle\sum_{e \in \delta(U)} x_e & \geq & 1 & \forall \text{ Steiner cut } U \\
 & x_e & \geq & 0 & \forall e \in E
\end{array}
$$

$\delta(U)$: Edges with exactly one endpoint in U.

Steiner trees: Dual LP

Dual LP has a variable y_U for all Steiner cuts U.

$$
\begin{array}{llccl}
\text{max} & \sum_{U} y_U & & & \\
\text{s.t.} & \sum_{U:\ e \in \delta(U)} y_U & \leq & c_e & \forall e \in E \\
 & y_U & \geq & 0 & \forall U
\end{array}
$$

$\delta(U)$: Edges with exactly one endpoint in U.

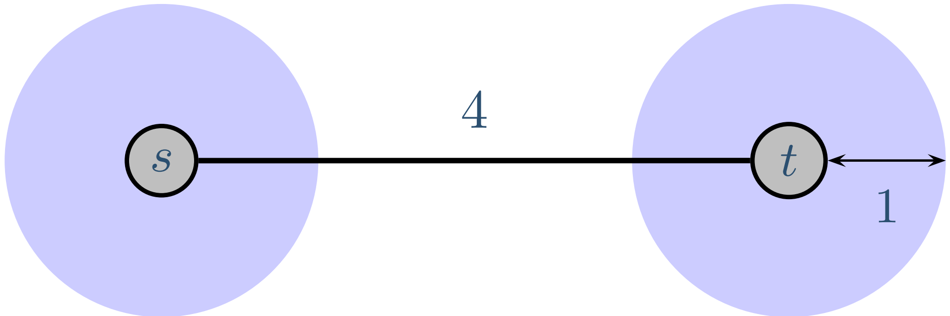
Dual LP: Pictorial View

- Can visualize y_U as disks around U with radius y_U. Example: Terminal pair $(s, t) \in R$, edge (s, t) with cost 4

$y_s = y_t = 0$

Dual LP: Pictorial View

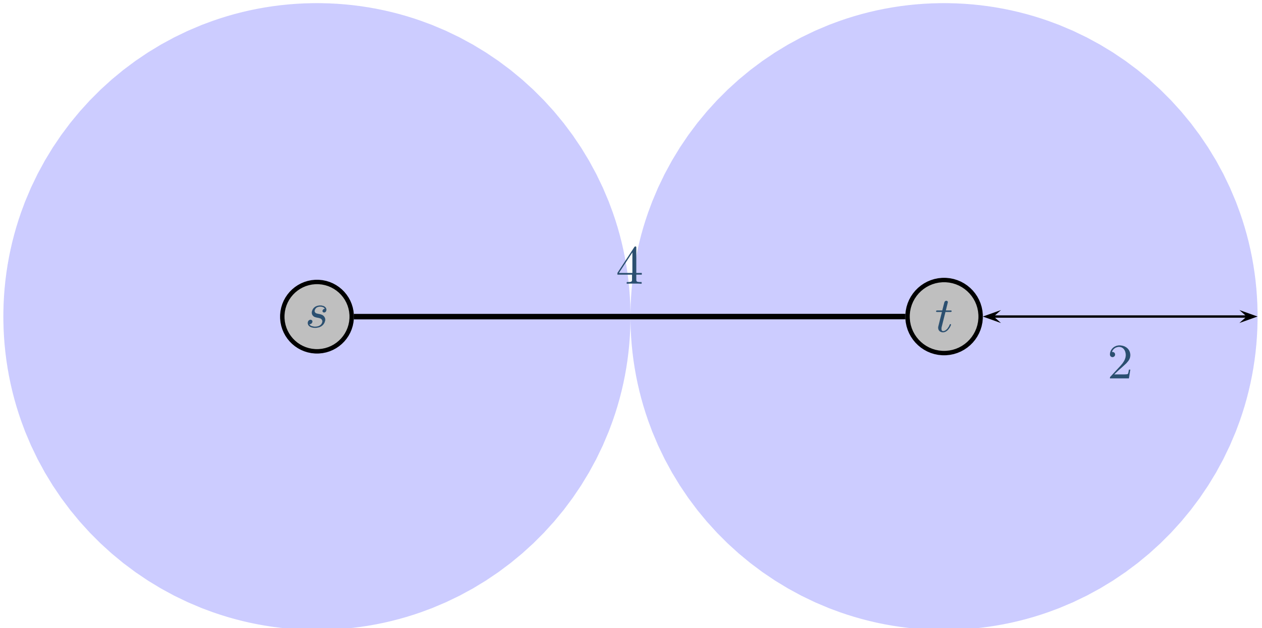
- Can visualize y_U as disks around U with radius y_U.
  Example: Terminal pair $(s, t) \in R$, edge (s, t) with cost 4



$$y_s = y_t = 1$$

Dual LP: Pictorial View

- Can visualize y_U as disks around U with radius y_U.
  Example: Terminal pair $(s,t) \in R$, edge (s,t) with cost 4

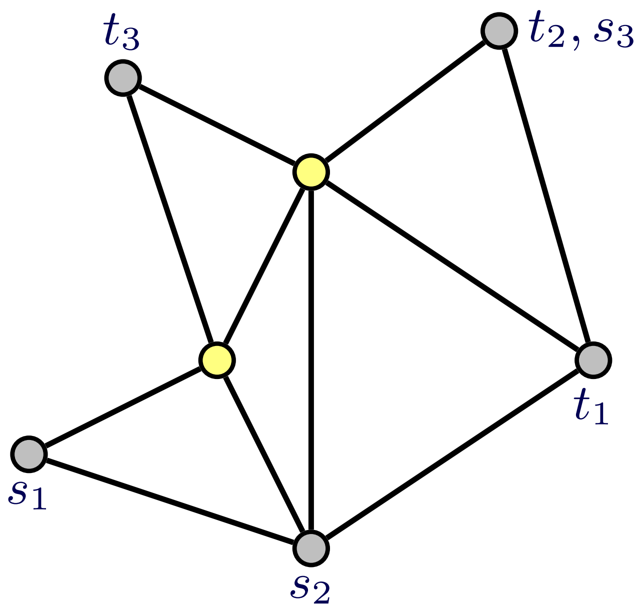


$y_s = y_t = 2$

Have: $y_s + y_t = 4 = c_{st}$. Edge (s,t) is tight.

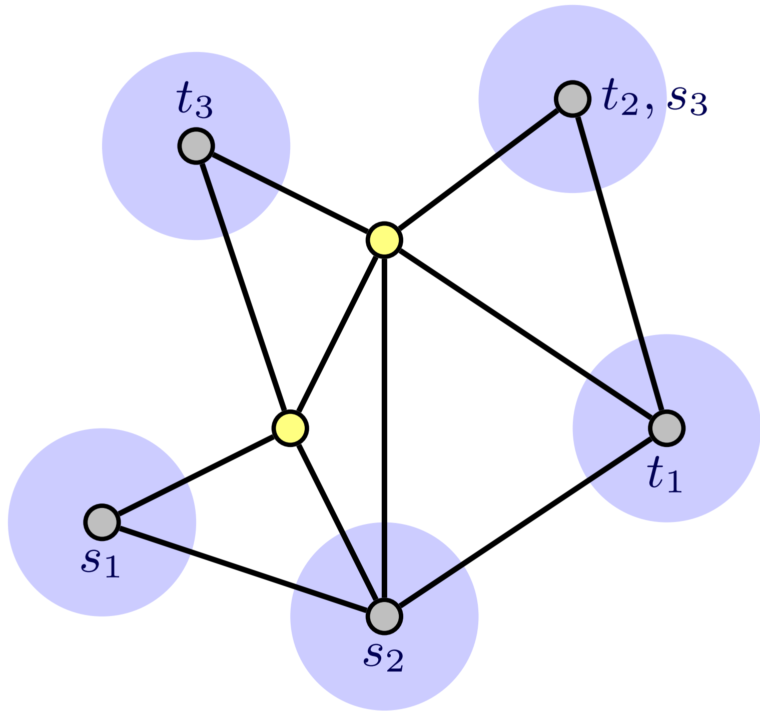
Algorithm SF: Example

Algorithm grows duals of connected components.



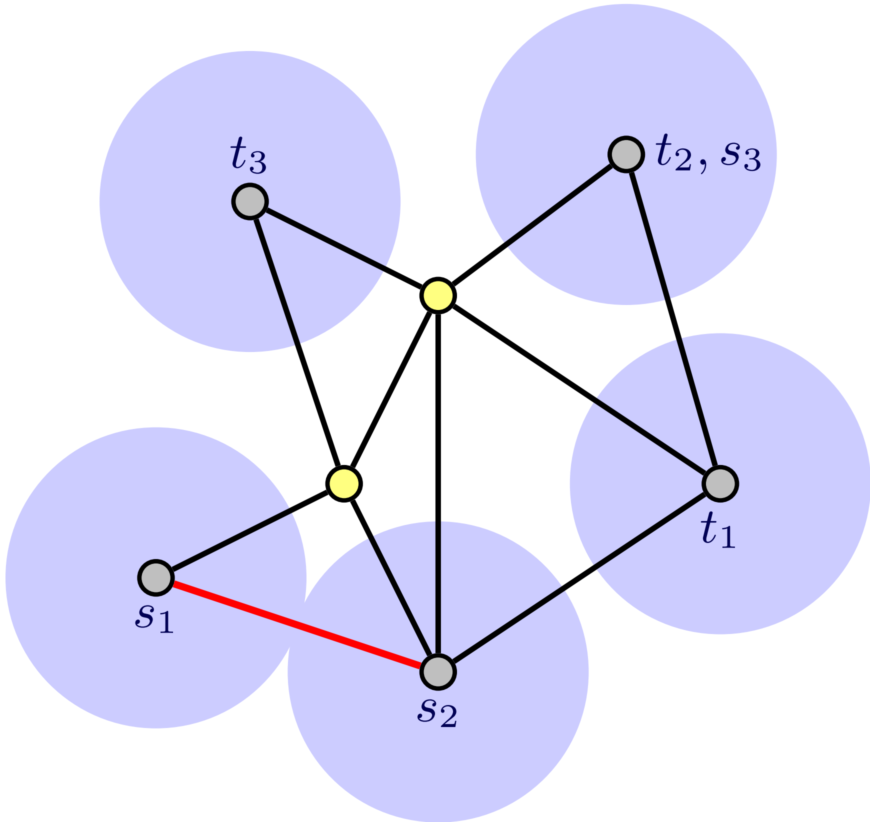
Algorithm SF: Example

Algorithm grows duals of connected components.



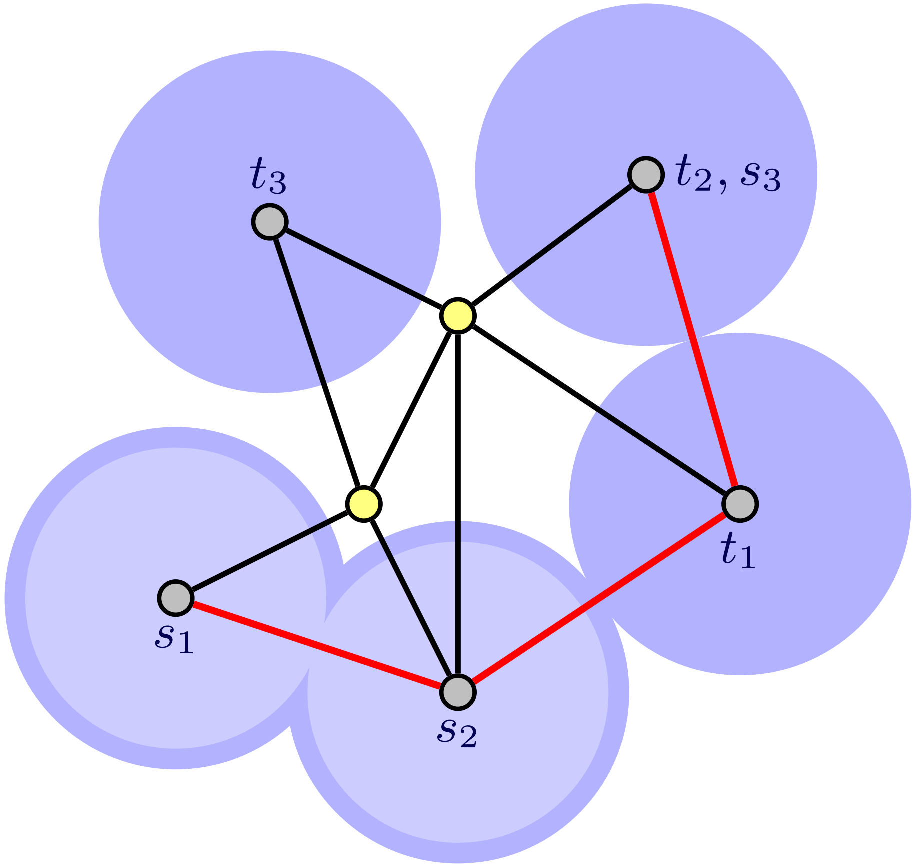
Algorithm SF: Example

Algorithm grows duals of connected components.

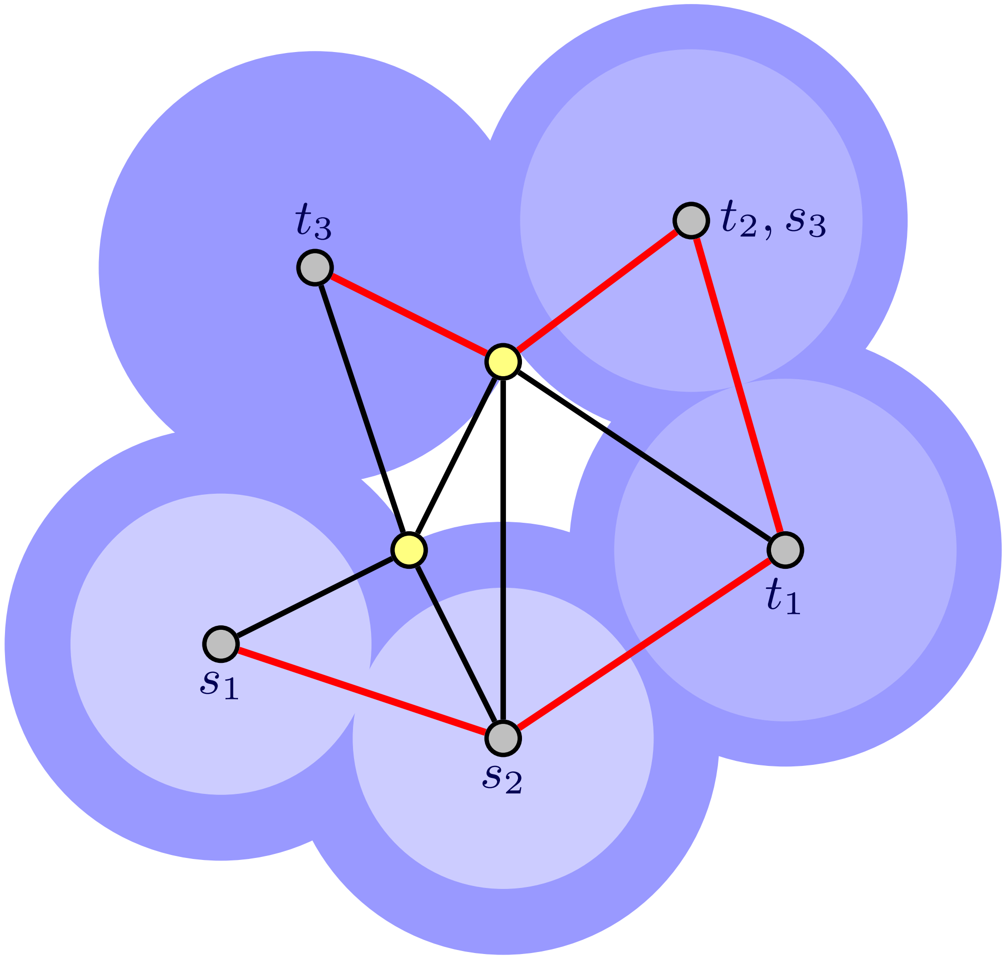


Algorithm SF: Example

Algorithm grows duals of connected components.



Algorithm grows duals of connected components.



PD-Algorithm: Properties

Theorem [Agrawal, Klein, Ravi '95]: Algorithm computes forest F and dual y such that

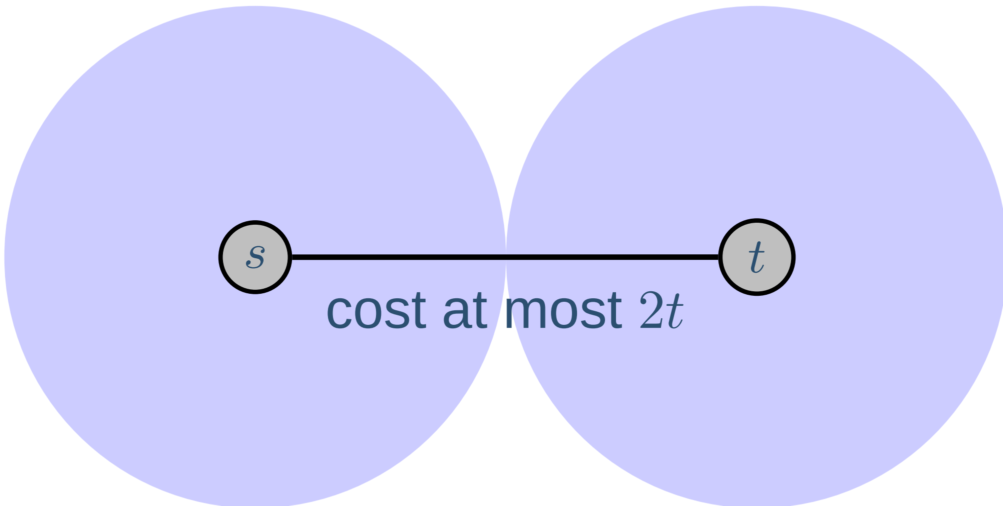
$$c(F) \leq (2 - 1/k) \cdot \sum_U y_U \leq (2 - 1/k) \cdot \mathtt{opt}_R.$$

PD-Algorithm: Properties

Theorem [Agrawal, Klein, Ravi '95]: Algorithm computes forest F and dual y such that

$$c(F) \leq (2 - 1/k) \cdot \sum_{U} y_U \leq (2 - 1/k) \cdot \texttt{opt}_R.$$

Main trick: Edge (s,t) becomes tight at time t.



Use twice the dual around s and t to pay for cost of path.

The AKR algorithm

Description of the algorithm

- Terminal t is active at time t if separated from its mate in the set of components active at time t
- A component is active at time t if it contains at least an active terminal
- The algorithm uniformly grows the dual variables for all maximal active components, i.e., those not contained in any other active component
- Whenever a path becomes tight, i.e. the dual constraints of all the edges of the path are tight, the two active components connected by the path are merged
- Let S_1 and S_2 the two merged component and let $S = S_1 \cup S_2$ be the resulting component. We stop raising the dual variables y_{S_1} and y_{S_2}. We start raising the dual variable y_S if S is active

Analysis of AKR for Steiner trees

- In the Steiner tree case one of the terminal vertices is denoted as the root of the tree and all other terminals need to connect to the root vertex
- In the Steiner tree case all terminal vertices are active till there is only one component including all the terminals.
- Let $\mathcal{U}_t$ be the set of active components at time t
- Let $F_t(S)$ be the tree spanning component $S \in \mathcal{U}_t$.
- **C**laim: The merging of two components at time t happens along a path of length at most $2t$ (Prove!)

Analysis of AKR for Steiner trees

Lemma: At any time t, for each component $S \in \mathcal{U}_t$:

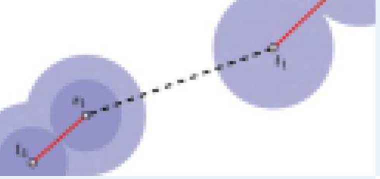
$$c(F_t(S)) \leq \sum_{U \subset S} 2y_U - 2t$$

. (Observe: $\subset$, not $\subseteq$!)

- *Basis of the induction.* The claim holds at time $t = 0$
- *Induction hypothesis.* Assume the claim holds for component S_1 formed at time t_1 and component S_2 formed at time t_2
- *Induction step.* At time $t \geq t_1, t_2$, components S_1 and S_2 merge to form $S = S_1 \cup S_2$
- The following relations holds at time t:

$$y_{S_1} = t - t_1 \text{ and } y_{S_2} = t - t_2$$

- The cost of the path connecting S_1 and S_2 is at most $2t$.



Analysis of AKR for Steiner trees

- *Induction step.* (contd)

$$
\begin{aligned}
c(F_t(S)) &\leq c(F_{t_1}(S_1)) + c(F_{t_2}(S_2)) + 2t \\
&\leq \sum_{U \subset S_1} 2y_U - 2t_1 + \sum_{U \subset S_2} 2y_U - 2t_2 + 2t \\
&= \sum_{U \subset S} 2y_U - 2y_{S_1} - 2y_{S_2} - 2t_1 - 2t_2 + 2t \\
&= \sum_{U \subset S} 2y_U - 2(t - t_1) - 2(t - t_2) - 2t_1 - 2t_2 + 2t \\
&= \sum_{U \subset S} 2y_U - 2t
\end{aligned}
$$

- Since $\sum_U y_U \leq c(OPT)$ and $c(OPT) \leq 2kt$ (largest cost of a solution since in the worst case $2k$ terminals are all active for a time t), the claim follows.

Analysis of AKR for Steiner trees

- An even simpler argument for the proof stems from proving by induction that every active component U holds t credits at time t.
- Two components merging at time t along a path of length at most $2t$ have $2t$ credits available:
  1. t credits are used to pay $\frac{1}{2}$ the cost of the connecting path
  2. t credits are given to the new component
- The solution is therefore half payed by the dual up to the final time of the algorithm

Analysis of AKR for Steiner Forest

We use the credit argument to prove the $2 - 1/k$ approximation of AKR for Steiner Forest

- In the execution of AKR for Steiner forest not all the components are active!
- Components are partitioned into Active and Inactive.
- A component that becomes inactive at time t retains t credits whereas the total dual inside the component pays half the cost of the tree
- A tight path connecting two active components may traverse an arbitrary number of inactive components
- The segments of the path traversing a component that became inactive at time t costs at most $2t$.
- The picture is actually a bit more complicated since inactive components are nested.

Analysis of AKR for Steiner Forest

- The two components that merge will bring $2t$ credits:
- We pay for a path that connects two active components as follows:
  1. t credits are used for paying the segments of the path that are outside the inactive components.
  2. t credits are given to the new component
  3. The credits of the inactive components are used to pay for half of the segments that traverse the inactive components
- We proved the $2 - 1/k$ approximation