Primal-Dual Algorithms for Network Design
The Steiner Tree problem

Stefano Leonardi

Sapienza Universitá di Roma

Spring 2010 – UPC, Barcelona

Talk Outline

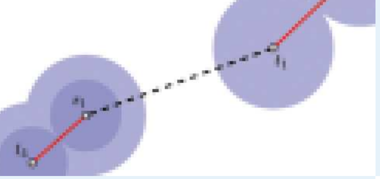
Part I	The Steiner tree problem
Parrt II	Primal-dual algorithms for Steiner problems
PartIII	The Steiner Forest problem

The Steiner Tree Problem

Steiner tree

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

- **S**teiner tree



Steiner tree

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

- **S**teiner tree
  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 tree

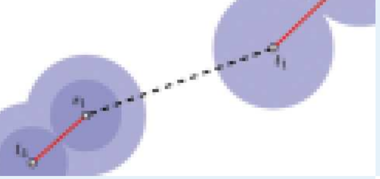
The Steiner tree problem has been defined for the first time by Gauss in a letter to Schumacher

- **S**teiner tree

  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \longrightarrow \mathbb{R}^+$;
  - 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 tree

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

- **Steiner tree**

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

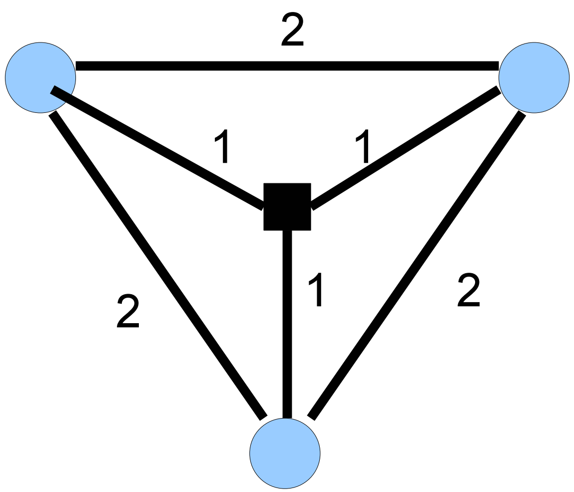
  Goal:

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

Steiner tree

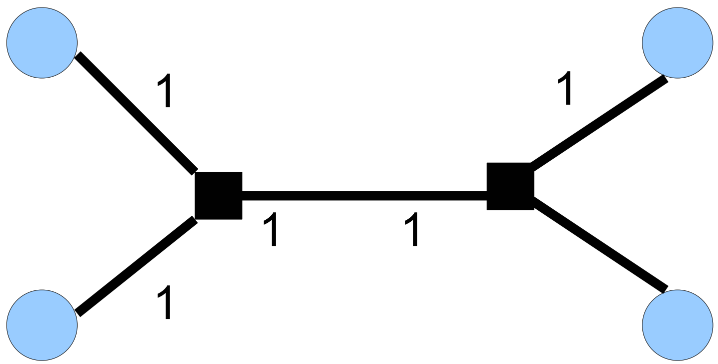
- We will consider the Steiner tree problem on metric spaces, i.e. $c(u,v) \leq c(u,w) + c(w,v)$.
- There exists a cost preserving reduction from Steiner tree to metric Steiner tree.
- Metric closure of G is the complete graph G' with costs $c'(u,v)$ equal to the shortest u,v path in G.
- We can transform 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 vertices may be strictly needed to design a connected spanning network
- Steiner vertice can be used for reducing the cost of the Spanning network:



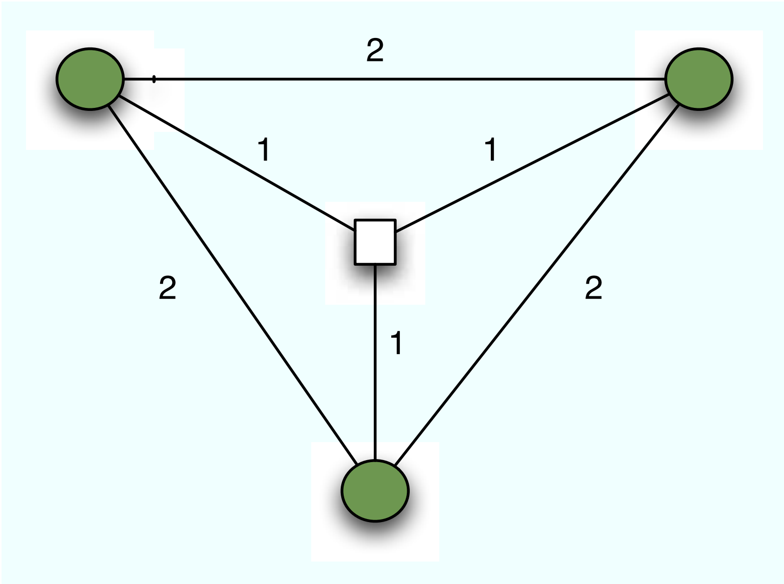
The optimal solution will include one Steiner vertex

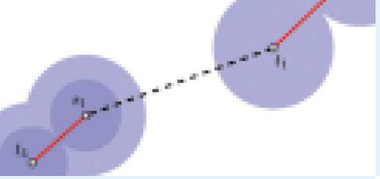
- Observe that this instance of Steiner tree is metric.
- Is this as crucial as in the Facility location problem?



The MST heuristic for 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 tree

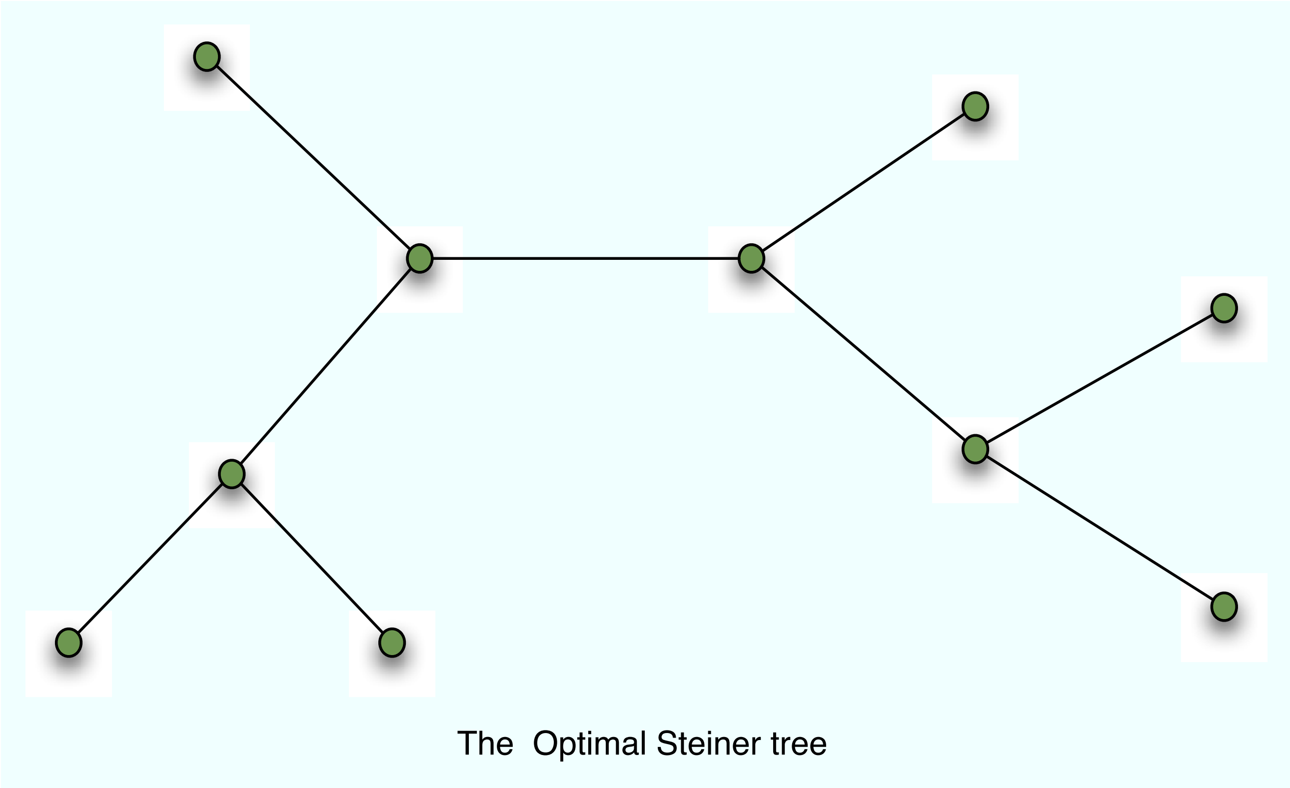
Theorem 1 *The cost of the MST on R in G' is at most twice the cost of the optimal Steiner tree of R in G.*

- 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
- Observe that 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:** the MST of R in G' is of cost at most $2 \times OPT$

The MST heuristic for Steiner tree

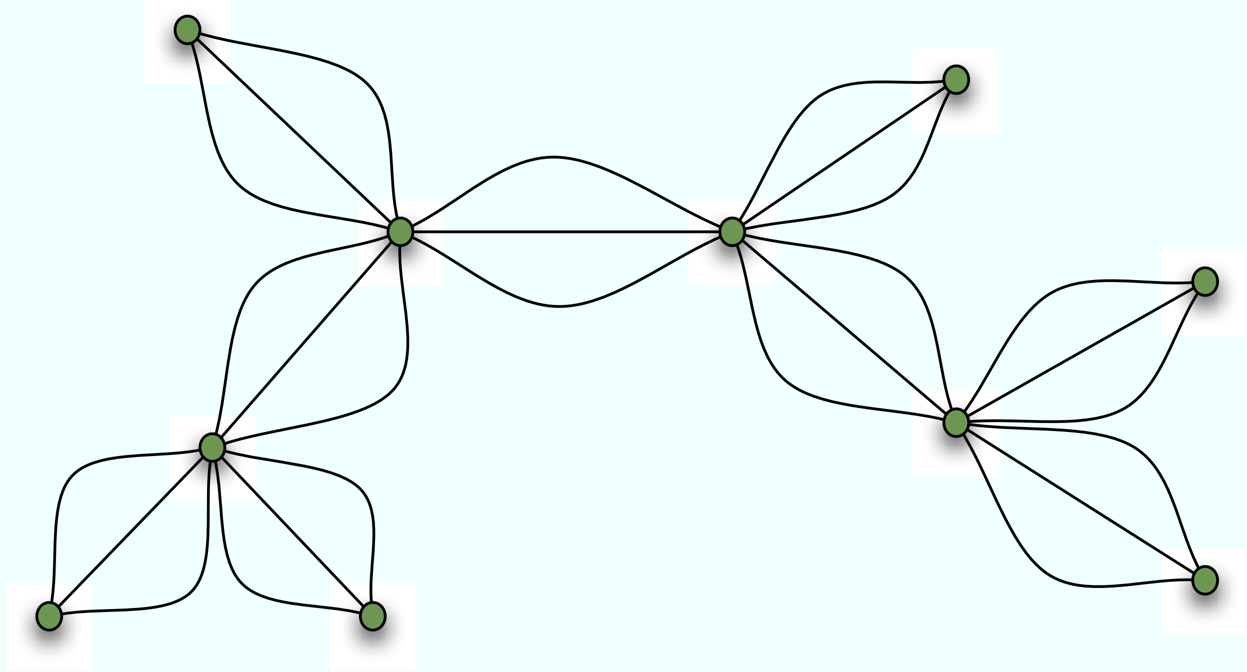
- The argument above shows that there exists a Spanning tree in G' of cost at most twice the optimal Steiner tree in G
- The MST in G' is therefore of cost smaller that twice the optimal Steiner tree in G
- The MST of R in G' can be transformed into a Steiner tree of R in G at no extra cost.

Start from the Optimal Steiner tree in G



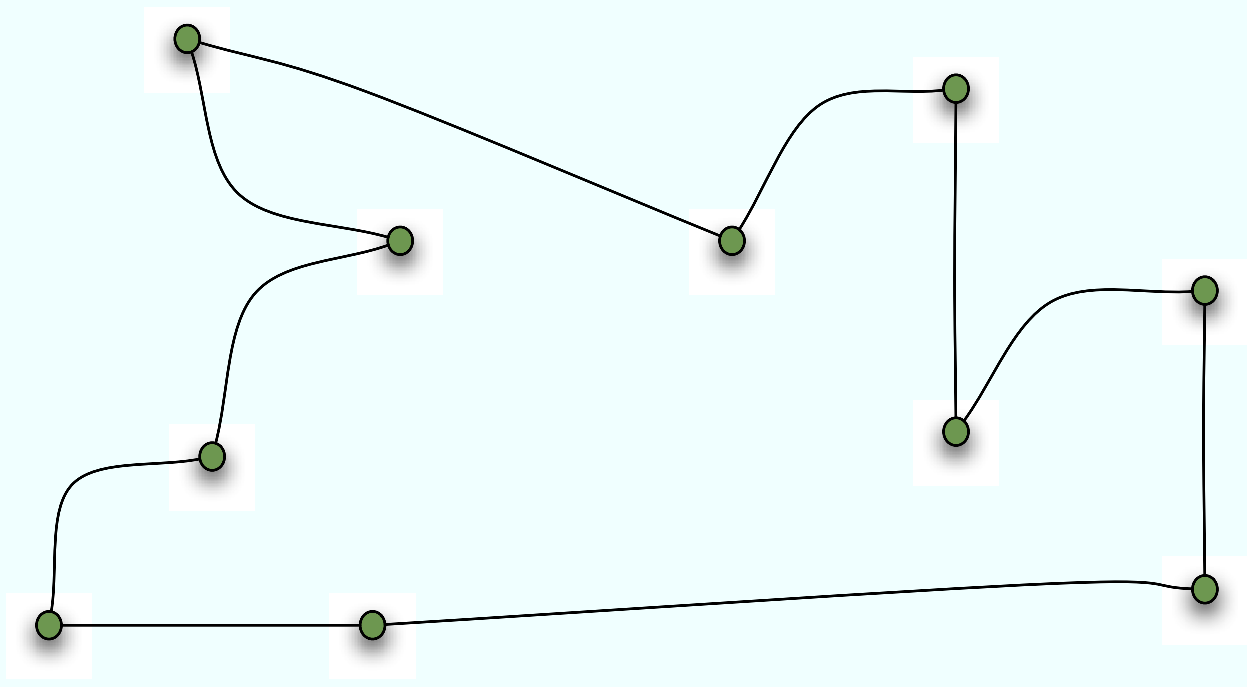
The MST heuristic for Steiner tree

Double the edges and obtain an Eulerian graphs

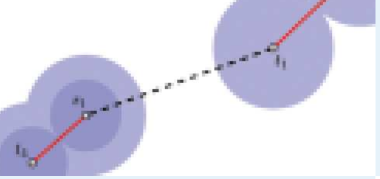


The Eulerian Tour

Find a Hamiltonian Path by Short-cutting the Eulerian tour



The Hamiltonian Cycle



The Steiner Forest Problem

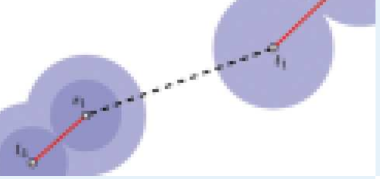
Steiner forests

- **S**teiner forests

Steiner forests

- **S**teiner forests

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



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 $(s, t) \in R$.

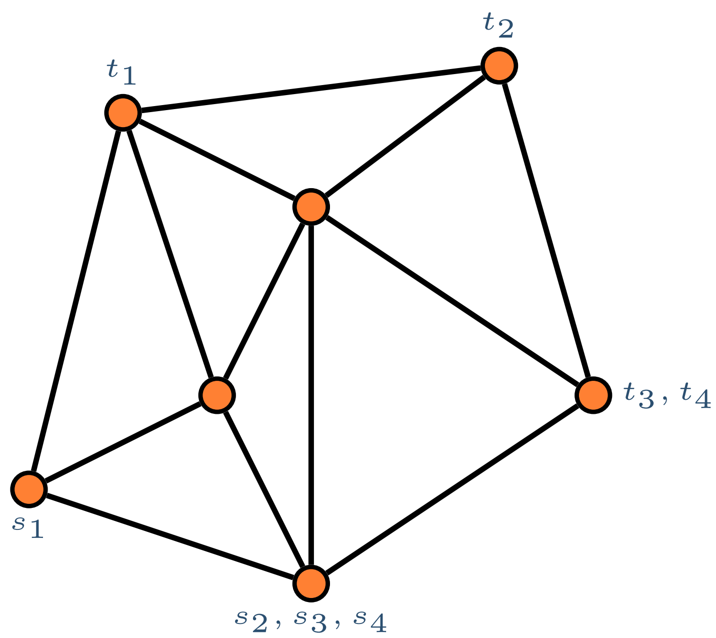
Steiner forests

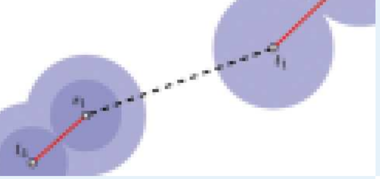
- **S**teiner forests
  Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \to \mathbb{R}^+$;
  - terminal-pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$.

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

Steiner forests: Example

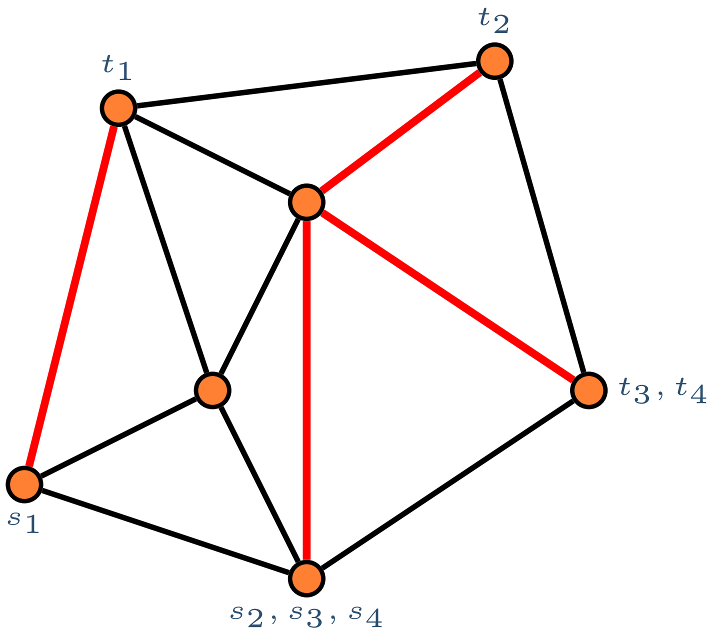
- Example with four terminal pairs: $R = \{(s_i, t_i)\}_{1 \leq i \leq 4}$
- All edges have unit cost.



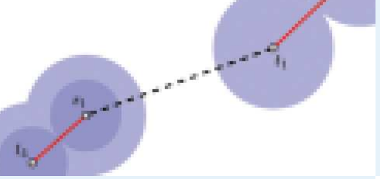


Steiner forests: Example

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



Total cost is 4!



Primal-dual algorithms for Steiner Forest

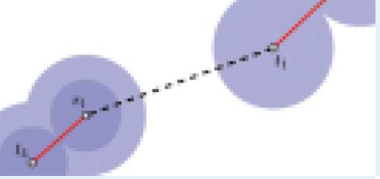
- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])

Primal-dual algorithms for Steiner Forest

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

Primal-dual algorithms for Steiner Forest

- [Agrawal, Klein, Ravi ’95] (see also [Goemans, Williamson ’95])
- The Goemans and Williamson algorithm is more general and applies to a wider set of network design problem
- We’ll present the AKR algorithm



Primal-dual algorithms for Steiner Forest

- [Agrawal, Klein, Ravi '95] (see also [Goemans, Williamson '95])
- The Goemans and Williamson algorithm is more general and applies to a wider set of network design problem
- We'll present the AKR algorithm

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.

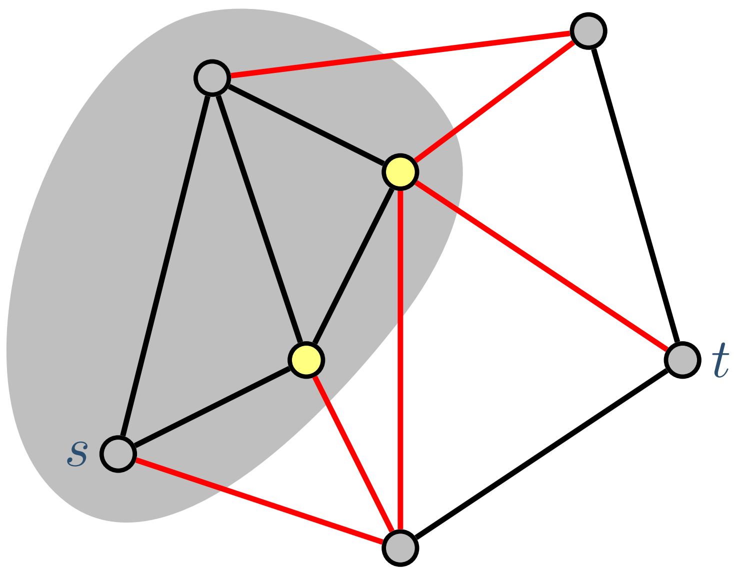
  Key trick: Use dual y and weak duality to bound cost of F.

Primal LP: Steiner Cuts

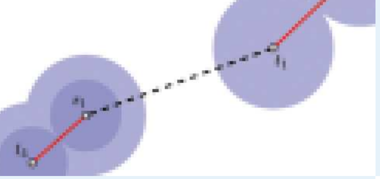
- Primal has variables x_e for all $e \in E$.
  $x_e = 1$ if e is in Steiner forest, 0 otherwise

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!

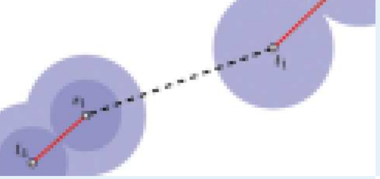


Primal LP: Steiner Cuts

Primal LP has one constraint for each Steiner cut.

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

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



Steiner trees: Dual LP

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

$$\max \sum_U y_U$$

$$\text{s.t.} \quad \sum_{U:\ e \in \delta(U)} y_U \leq c_e \quad \forall e \in E$$

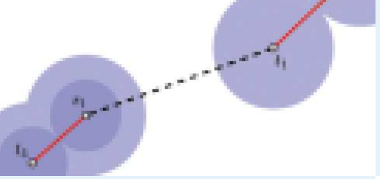
$$y_U \geq 0 \quad \forall U$$

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

Dual LP: Pictorial View

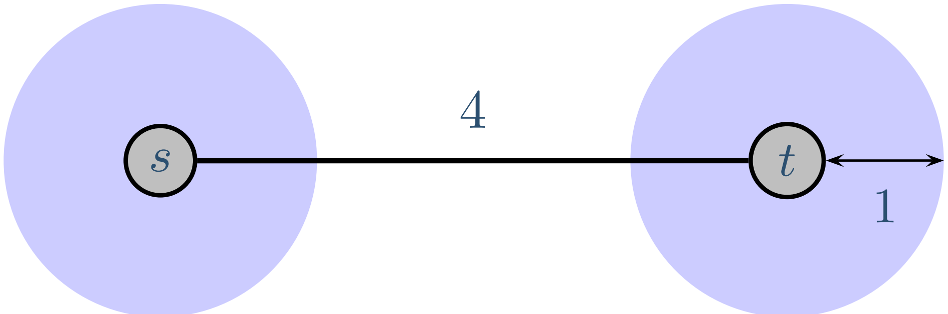
- Can visualize y_U as disks around U with radius y_U.
  Example: Terminal pair $(s, t) \in R$, edge (s, t) with cost 4

$y_s = y_t = 0$

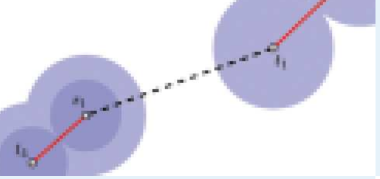


Dual LP: Pictorial View

- 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

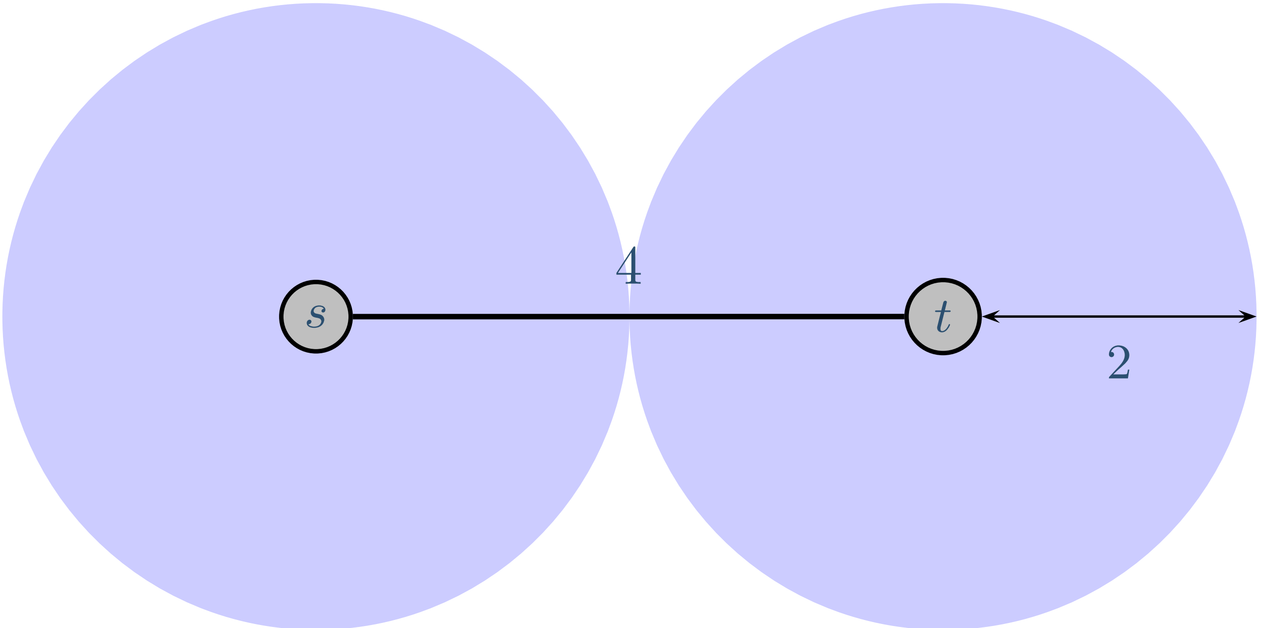


$y_s = y_t = 1$



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

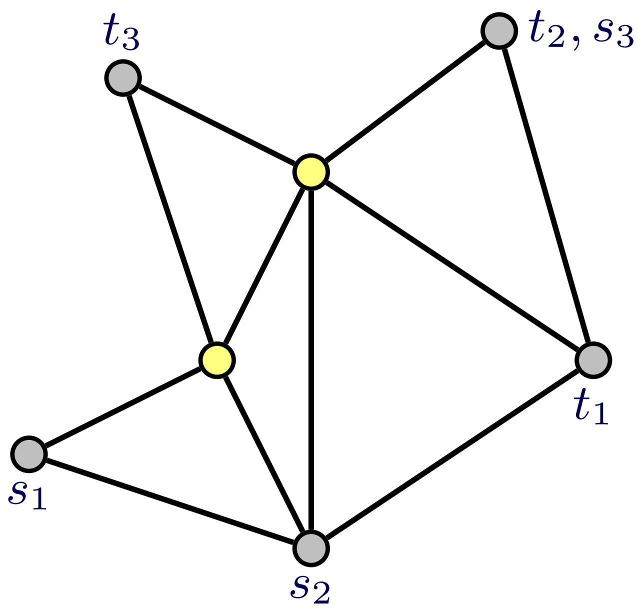


$y_s = y_t = 2$

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

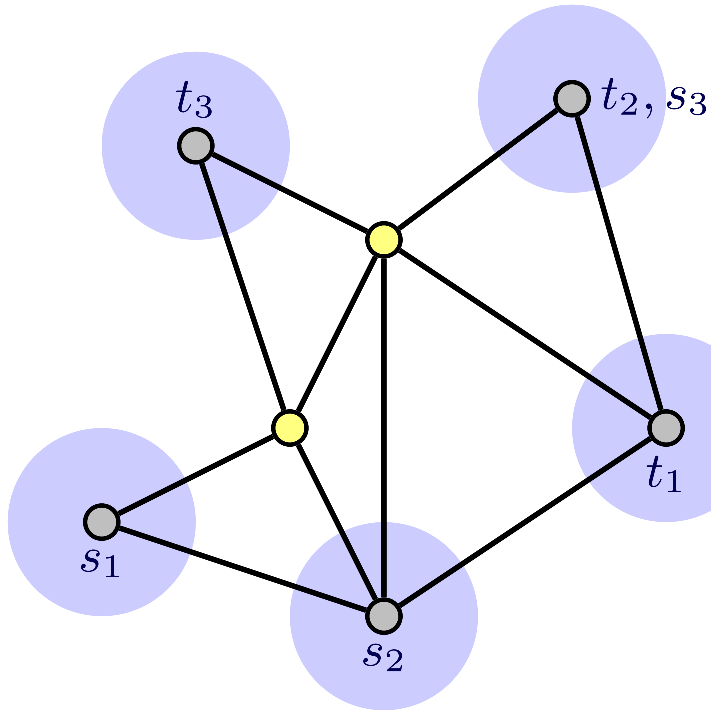
Algorithm SF: Example

Algorithm grows duals of connected components.

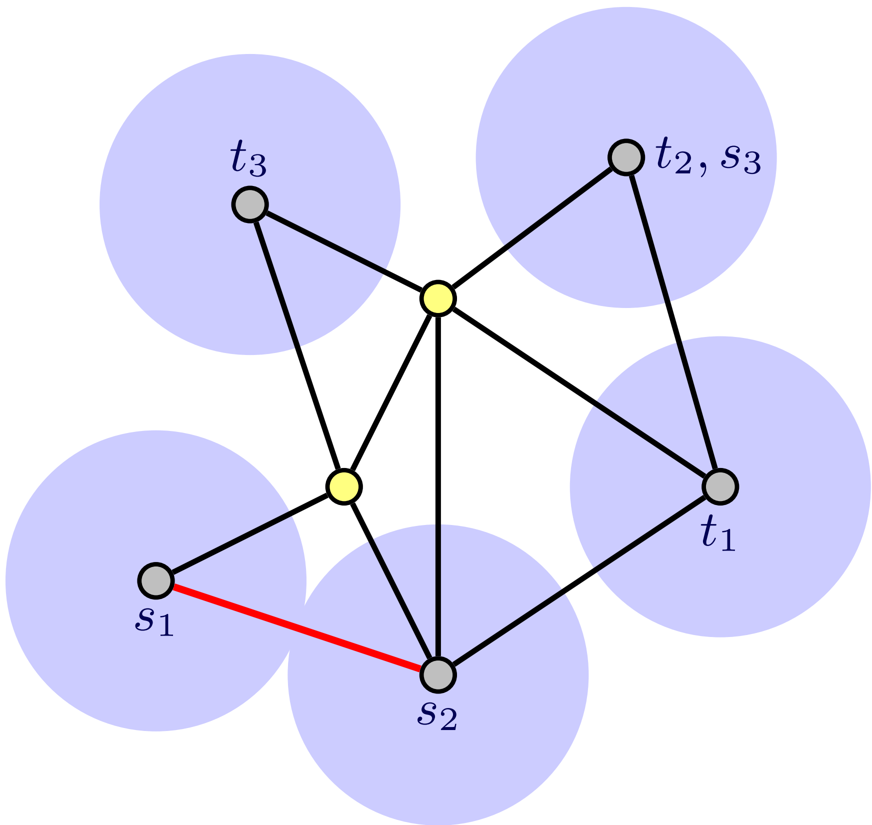


Algorithm SF: Example

Algorithm grows duals of connected components.

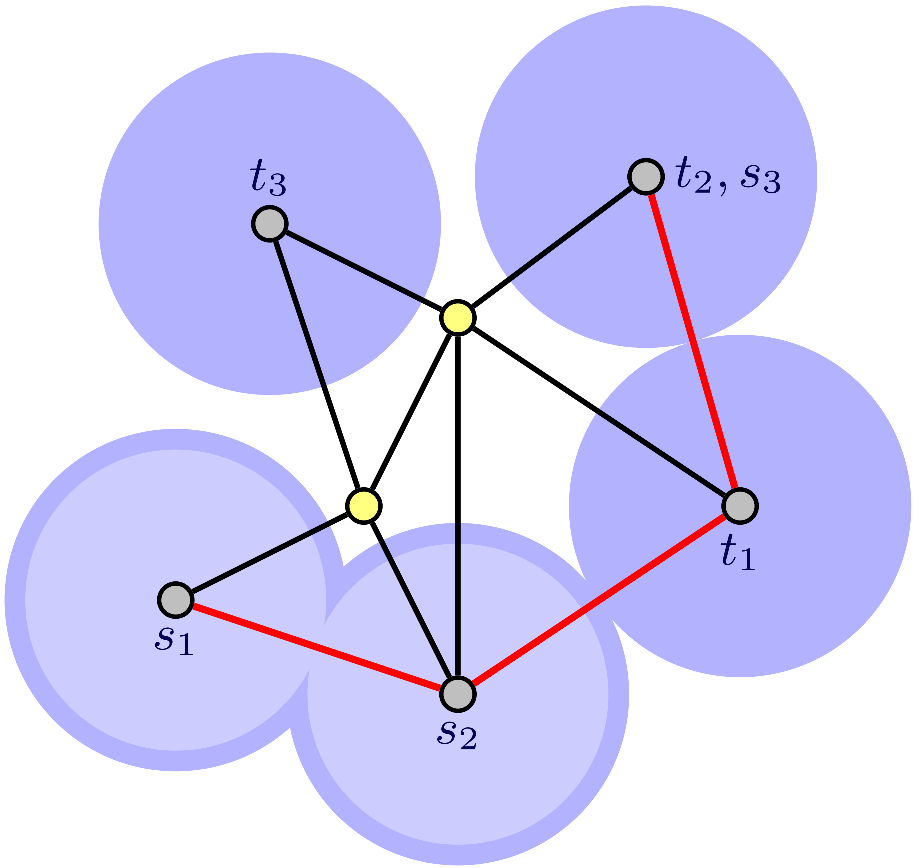


Algorithm grows duals of connected components.



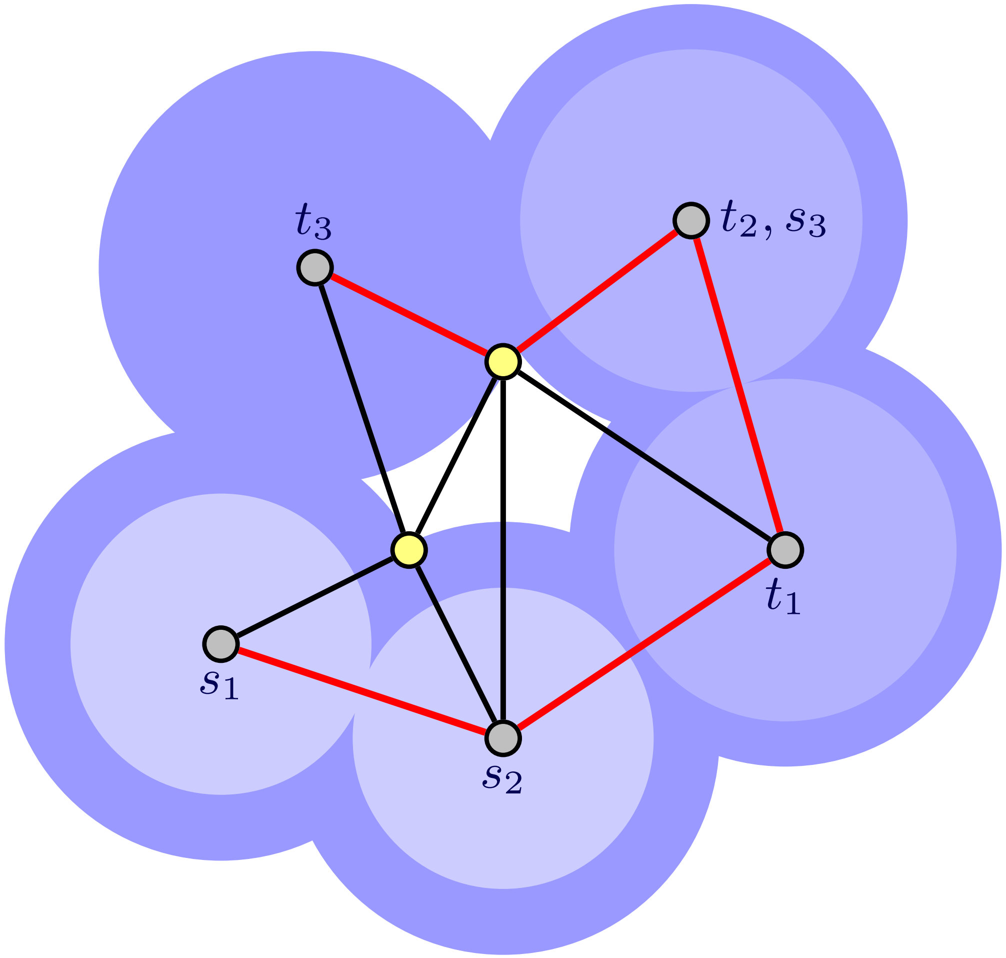
Algorithm SF: Example

Algorithm grows duals of connected components.



Algorithm SF: Example

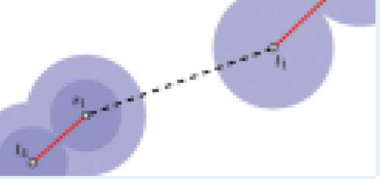
Algorithm grows duals of connected components.



PD-Algorithm: Properties

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

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

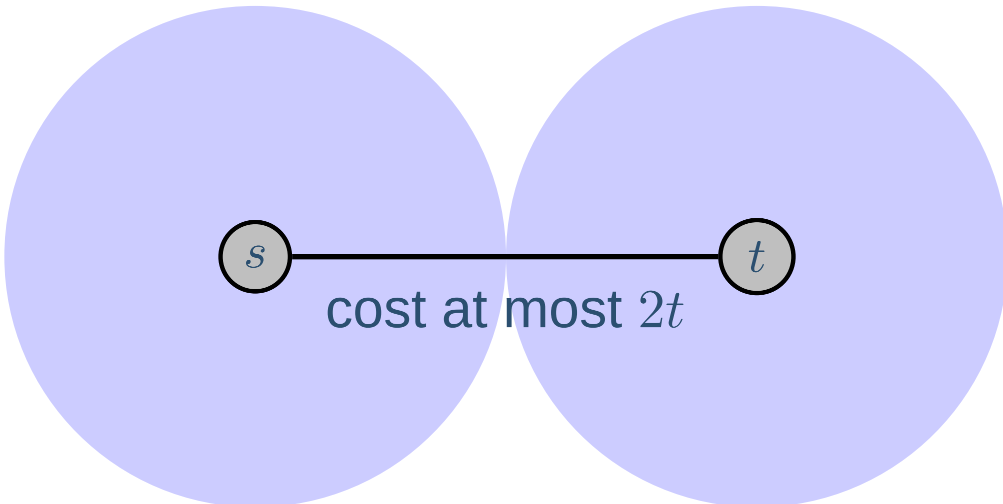


PD-Algorithm: Properties

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

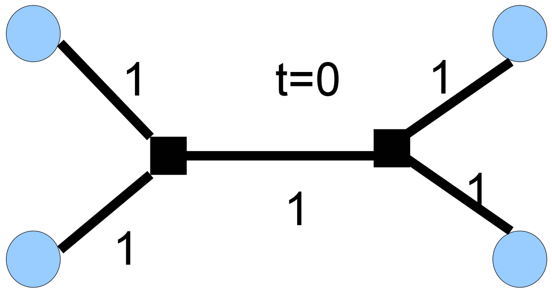
- Terminal t is active at time t if separated from its mate in the set of active components at time t
- A component is active at time t if it contains at least an active terminal
- The algorithm uniformly grows the dual variables for all maximal active components, i.e., those not contained in any other 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

Tight paths

- A path becomes tight when all edges of the paths are tight.
- We restrict consideration to path between active terminals.
- This does not prevent the existence of Steiner vertices along the path
- However, we can restrict to tight paths that do not traverse active terminals
- We can otherwise claim the existence of shorter path that becomes tight at the same time

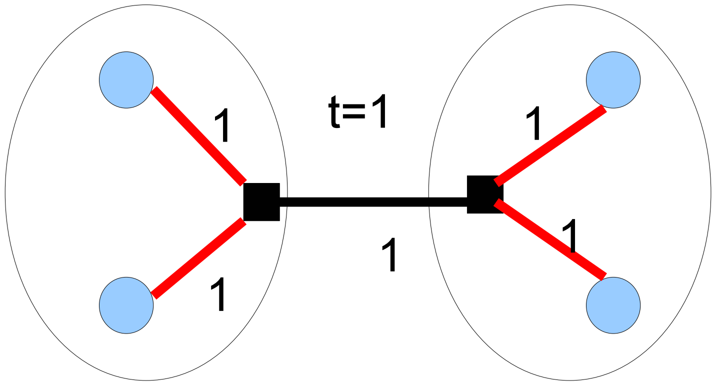
Tight paths

Steiner tree instance.
4 components active at time 0.



Tight paths

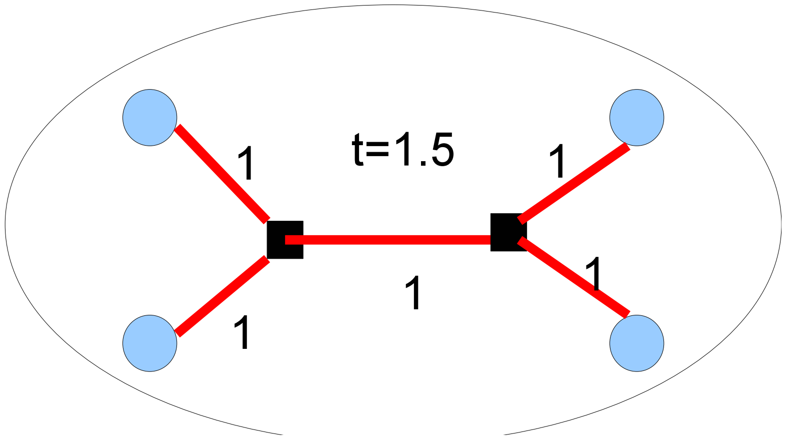
At time 1 two paths become tight. Two components are active (only 2 dual variables grow).



Tight paths

At time 1.5 a new path becomes tight.

Total dual collected = 3



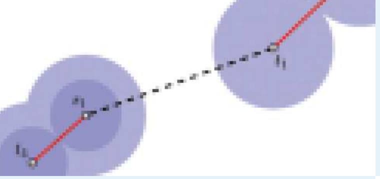
Analysis for Steiner trees

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 $S \in \mathcal{U}_t$.

Analysis of AKR for Steiner trees

- **C**laim: The merging of two components at time t happens along a path of length at most $2t$
- The path connects two active terminals along a set of Steiner vertices.
- The path has been loaded by at most two active components at any time $t' \leq t$
- Indeed, the path cannot be loaded by more than two active components, otherwise there would be at least one terminal on the path
- It follows that the length of the path is at most 2t.



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:

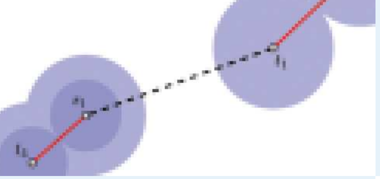
$$y_{S_1} = t - t_1 \text{ and } y_{S_2} = t - t_2$$

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

Analysis of AKR for Steiner trees

- *Induction step.* (contd)

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



Analysis of AKR for Steiner trees

We proved:

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

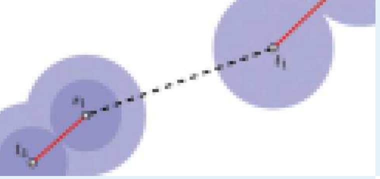
We know:

- $\sum_U y_U \leq c(OPT)$
- $c(OPT) \leq 2kt$:
  - There must exists at least two terminals at distance at least $2t$
  - If the process runs for t units of time, the sum of the distances between the the k terminal pairs is at most $2t$.

Analysis of AKR for Steiner trees

We conclude:

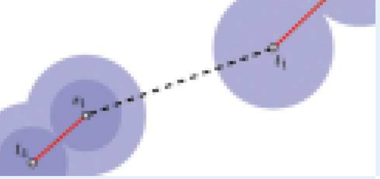
$$\begin{aligned} c(F_t(S)) &= \sum_{U \subset S} 2y_U - 2t \\ &\leq 2c(OPT) - 2\frac{c(OPT)}{2k} \\ &= \left(2 - \frac{1}{k}\right) c(OPT) \end{aligned}$$



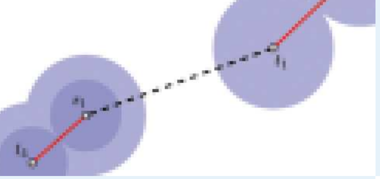
Analysis of AKR for Steiner trees

A very intuitive argument for the proof

- We prove by induction that every active component U holds t credits at time t that comes from a feasible dual
- 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 for Steiner Forest

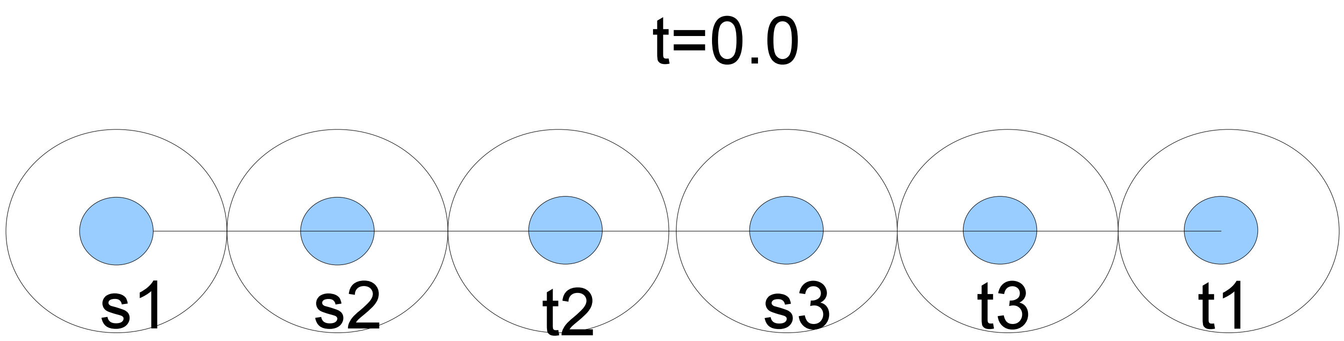


The Analysis for Steiner Forest

- In the execution of AKR for Steiner forest not all the components are active!
- Components are partitioned into Active and Inactive
- Inactive components do not have inside any active terminal
- Two major arguments in the analysis of Steiner tree breaks:
  - The cost of constructing a tight path at time t may be much bigger than 2t
  - There could be several *inactive* terminal vertices on a tight path.

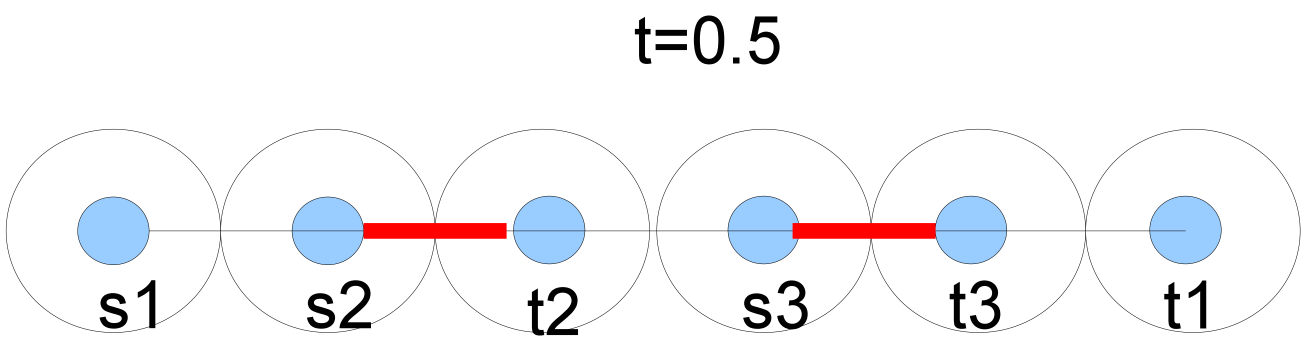
The analysis for Steiner Forest

Consider the following example with all edges of length 1 apart from the edges (s_1, s_2) and (t_3, t_1) that are of lengths $1 + \epsilon$.

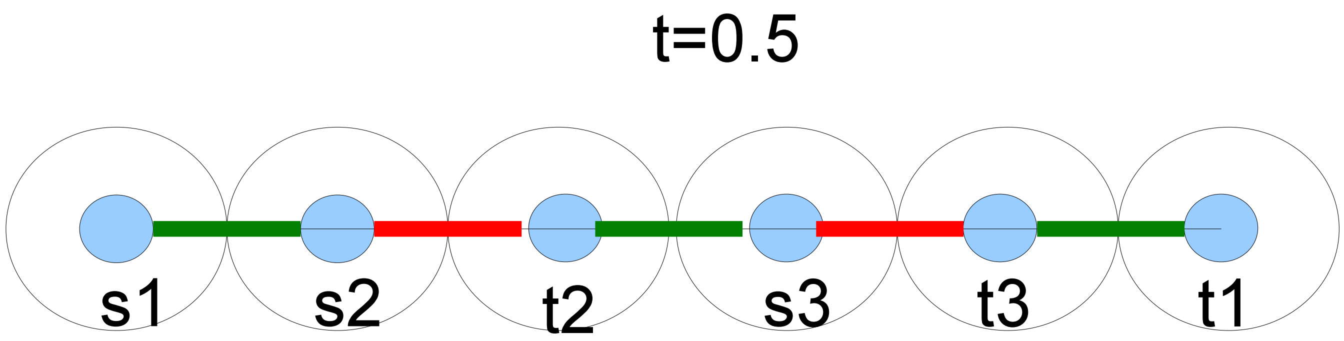


The analysis for Steiner Forest

- At time 0.5 the two internal pairs form two inactive components
- The dual variables of the two components stop growing



- At time $0.5+\epsilon$ the path connecting s_1 to t_1 becomes tight.
- The path has cost $3+2\epsilon$
- How do we pay for this path?

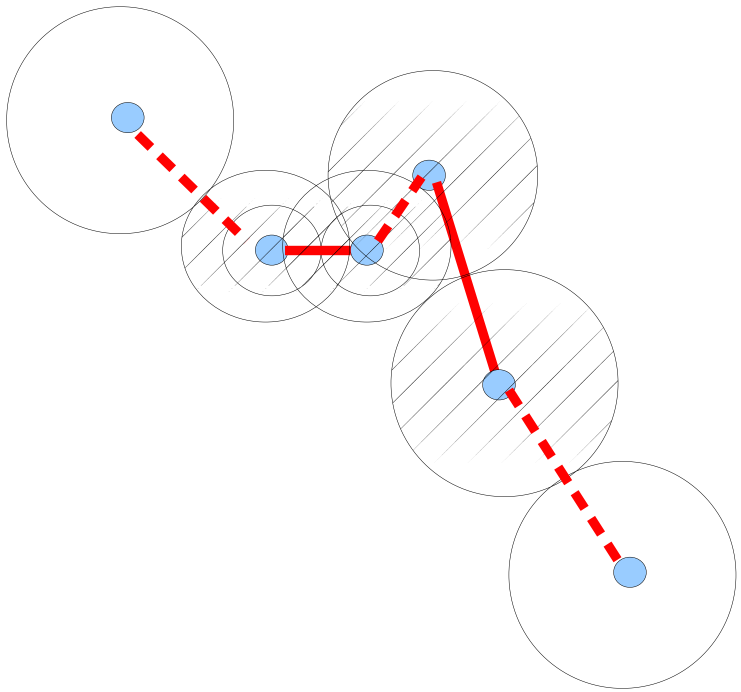


Analysis of AKR for Steiner Forest

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

- A component that becomes inactive at time t retains t credits
- 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.
- We need therefore to apply this argument recursively for any inactive component traversed by the path.

Example of nested components:



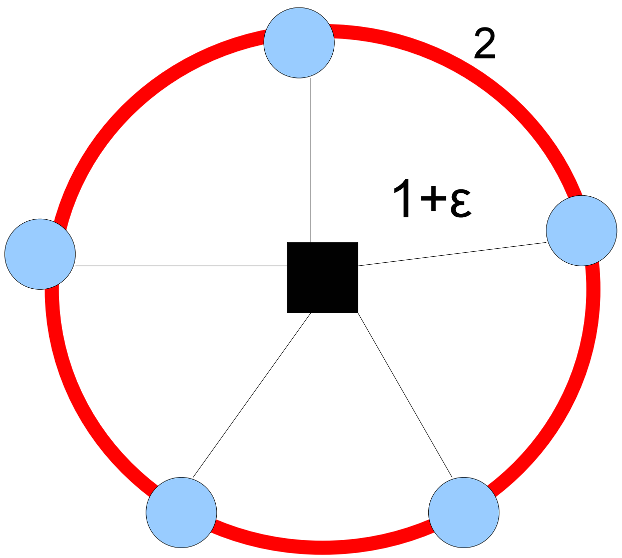
Analysis of AKR for Steiner Forest

- 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

Integrality gap for the LP relaxation

The line example has integrality gap 2 but still the solution found is optimal. The following example of Steiner tree has

integrality gap 2 but also provides a solution that is 2-approximated



The Goemans and Williamson Algorithm

- It is also a Primal-dual algorithm with approach similar to AKR
- It buys edges one by one even if they are not on tight paths between active terminals
- Many edges that are bought are useless and needs to be removed from the solution in a final *reserve pruning* stage
- The GW algorithm has also many other applications
- Proof for Steiner forest is slightly more involved