# Primal-Dual Approximation Algorithms for Network Design

Philip Hodges

Department of Combinatorics and Optimization
University of Waterloo

Spring 2018

The description of the algorithm in this presentation follows the exposition in the text, "The Design of Approximation Algorithms" by David P. Williamson and David B. Shmoys. Several of the graphics used also appear in this text.

# Table of Contents

# Table of Contents

# Generalized Steiner Tree Problem

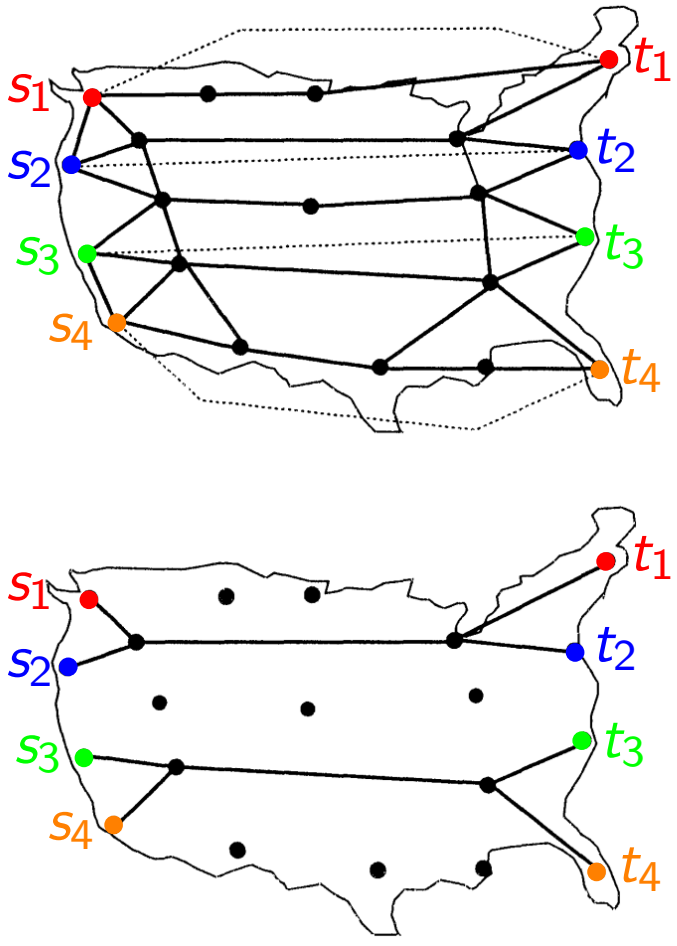
Given a graph $G = (V, E)$ with nonnegative edge costs $c_e$ for $e \in E$ and pairs of vertices $s_i, t_i \in V$, find a minimum-cost subset of edges $F \subseteq E$ such that every pair $(s_i, t_i)$ is connected in $(V, F)$.

# Generalized Steiner Tree Problem

Given a graph $G = (V, E)$ with nonnegative edge costs $c_e$ for $e \in E$ and pairs of vertices $s_i, t_i \in V$, find a minimum-cost subset of edges $F \subseteq E$ such that every pair $(s_i, t_i)$ is connected in $(V, F)$.

We call the vertices that are part of some requirement pair *terminals*.

# Example



# Is the problem hard?

- The Generalized Steiner Tree Problem is a generalization of the Steiner Tree Problem.
- The Steiner Tree Problem is one of the original 21 problems shown to be NP-complete by Richard Karp (1972).

# Is approximation easy?

Let's propose a simple algorithm:

# Is approximation easy?

Let's propose a simple algorithm:

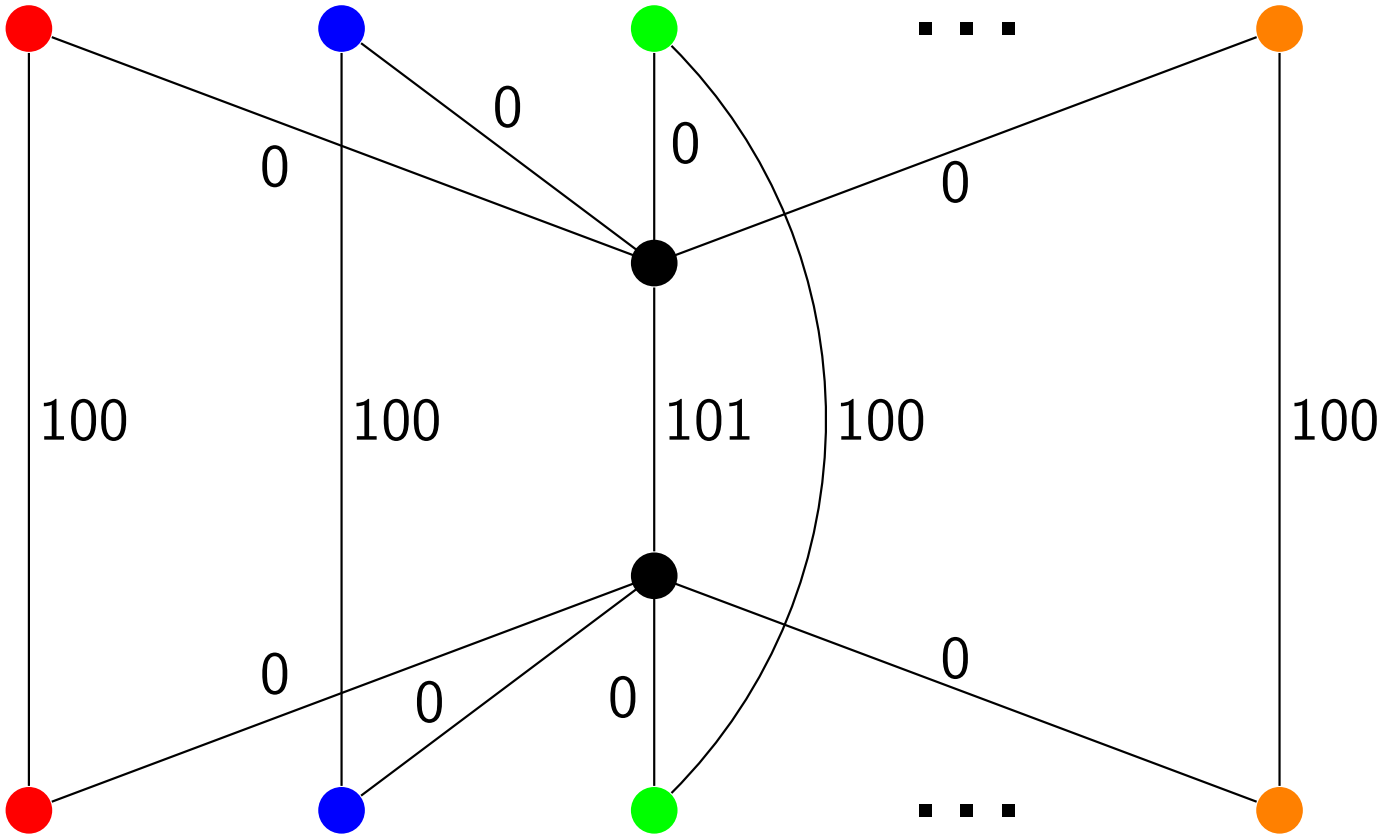
1. For each terminal pair $(s_i, t_i)$, find the shortest path $P_i$ in $G$ between $s_i$ and $t_i$.

# Is approximation easy?

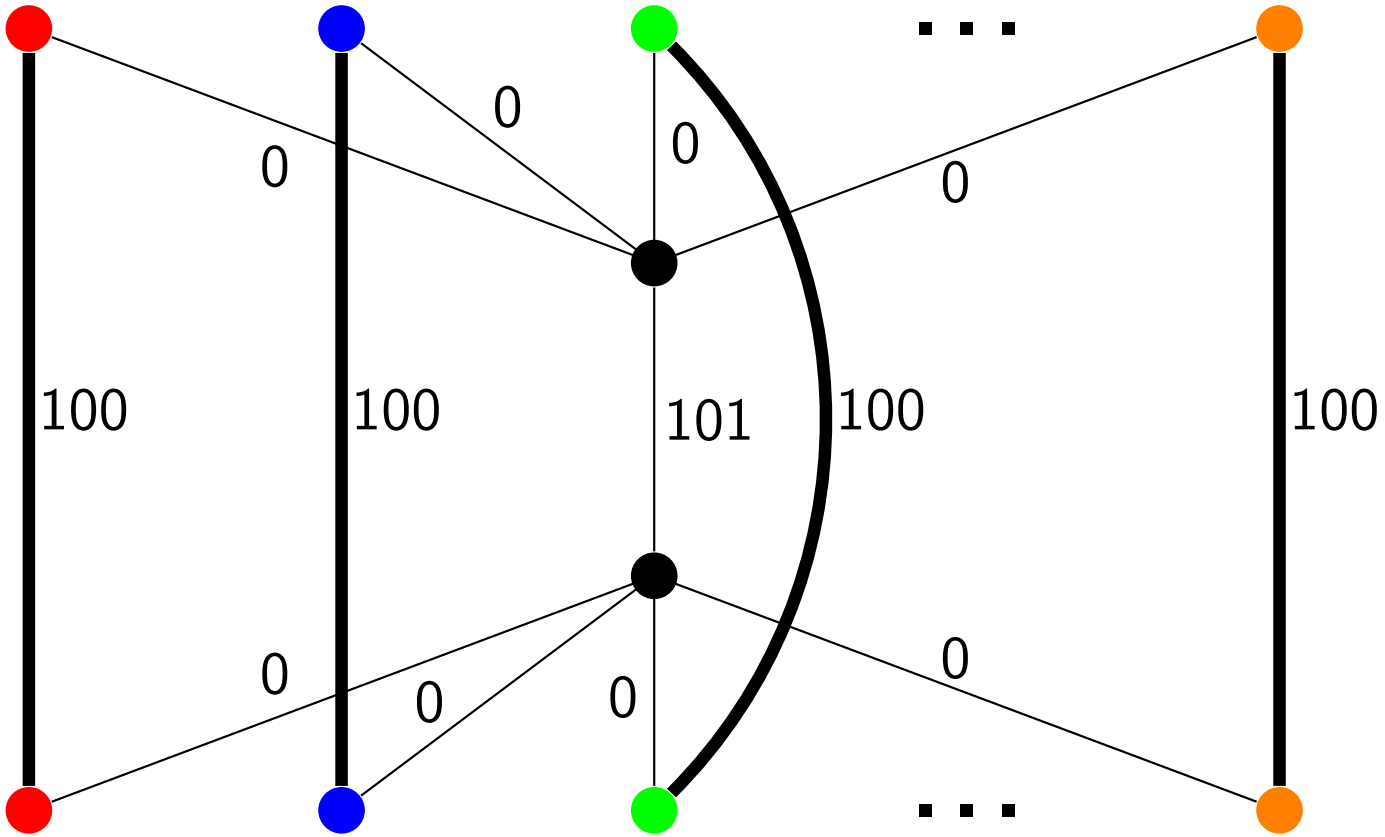
Let's propose a simple algorithm:

1. For each terminal pair $(s_i, t_i)$, find the shortest path $P_i$ in $G$ between $s_i$ and $t_i$.
2. Take the output edge set $F$ to be the union of all paths $P_i$.

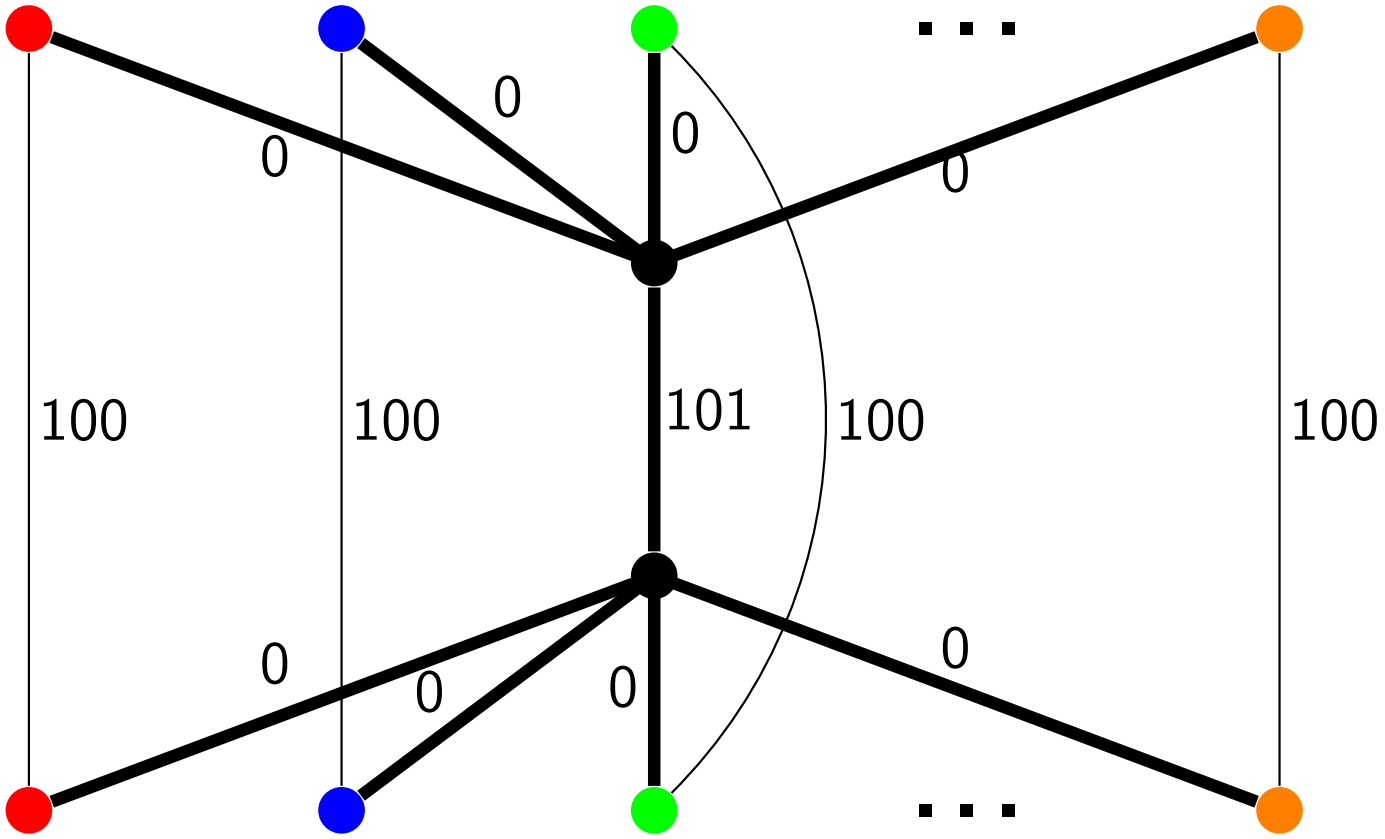
# Bad Example



# Bad Example



# Bad Example

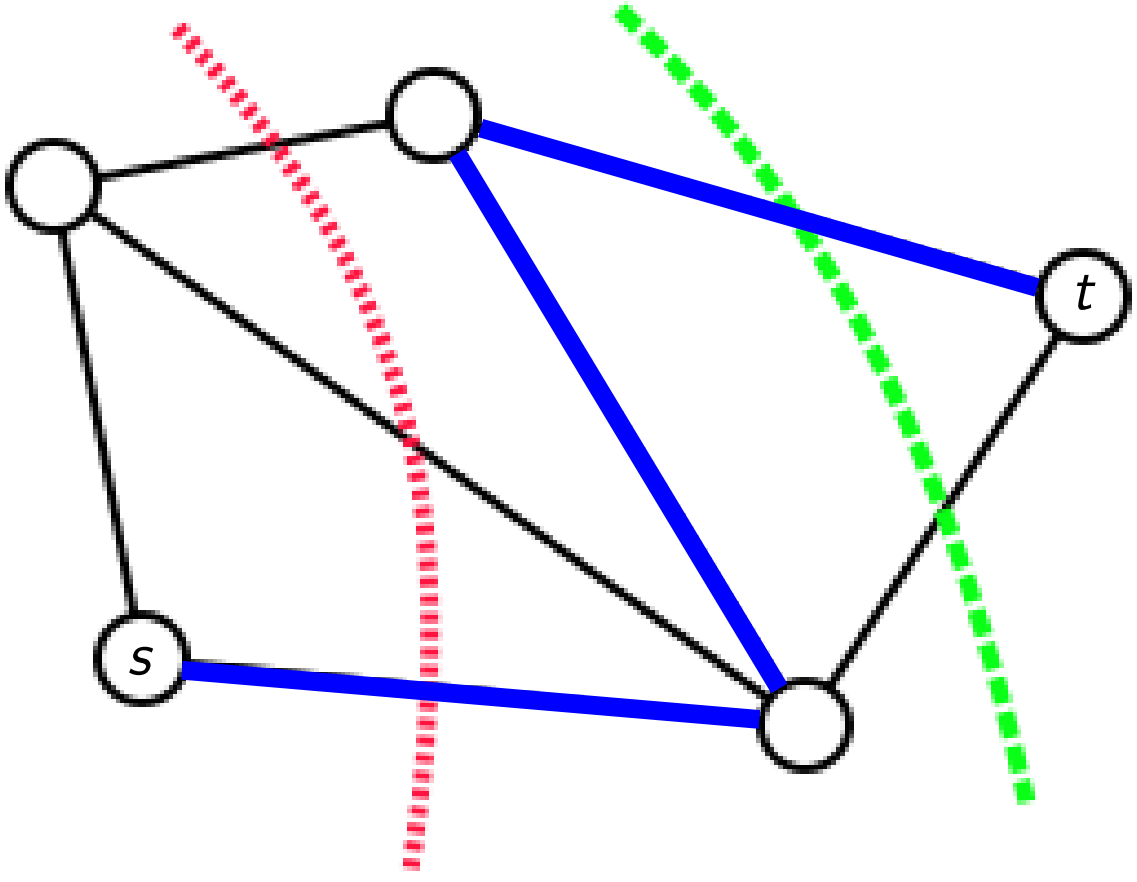


# Table of Contents

# Formulating a linear program

If we require that our solution contains at least one edge from each edge set separating a requirement pair, then the requirement pairs will be connected by at least one path, by the max-flow min-cut theorem.

# Max-flow Min-cut



# LP Formulation

Let us consider the case of only one pair of terminals $s, t$.

**Primal ($P$)**

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S : s \in S, t \notin S \\ & x_e \geq 0, \quad \forall e \in E \end{aligned}$$

**Dual ($D$)**

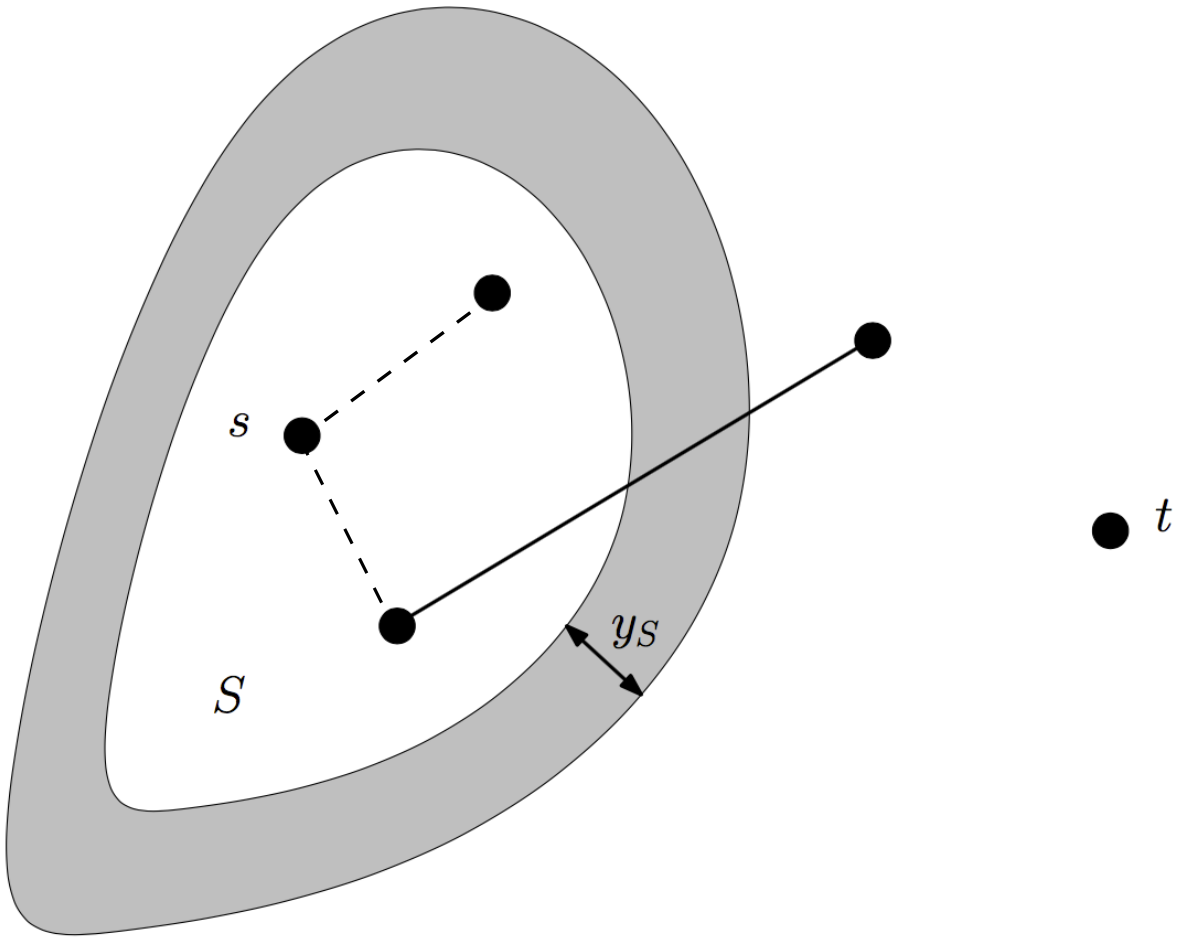
$$\begin{aligned} \text{maximize} \quad & \sum_{S} y_S \\ \text{subject to} \quad & \sum_{S : e \in \delta(S)} y_S \leq c_e, \quad \forall e \in E \\ & y_S \geq 0, \quad \forall S : s \in S, t \notin S \end{aligned}$$

But this is simply a formulation of the shortest path problem!

# Dijkstra's Algorithm

One way to describe Dijkstra's algorithm for finding a shortest path is by growing a moat around the current connected component, and selecting the next edge based on which vertex is first touched by the moat.

# Dijkstra's Illustration



# LP Formulation

Let $\hat{S}$ be the set of all subsets of $V$ separating some pair $s_i, t_i$.
That is, $\hat{S} = \{S \subseteq V : \exists i \ \ |S \cap \{s_i, t_i\}| = 1\}$.

Primal $(P)$

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S \in \hat{S} \\ & x_e \geq 0, \quad \forall e \in E \end{aligned}$$

Dual $(D)$

$$\begin{aligned} \text{maximize} \quad & \sum_{S \in \hat{S}} y_S \\ \text{subject to} \quad & \sum_{S : e \in \delta(S)} y_S \leq c_e, \quad \forall e \in E \\ & y_S \geq 0, \quad \forall S \in \hat{S} \end{aligned}$$

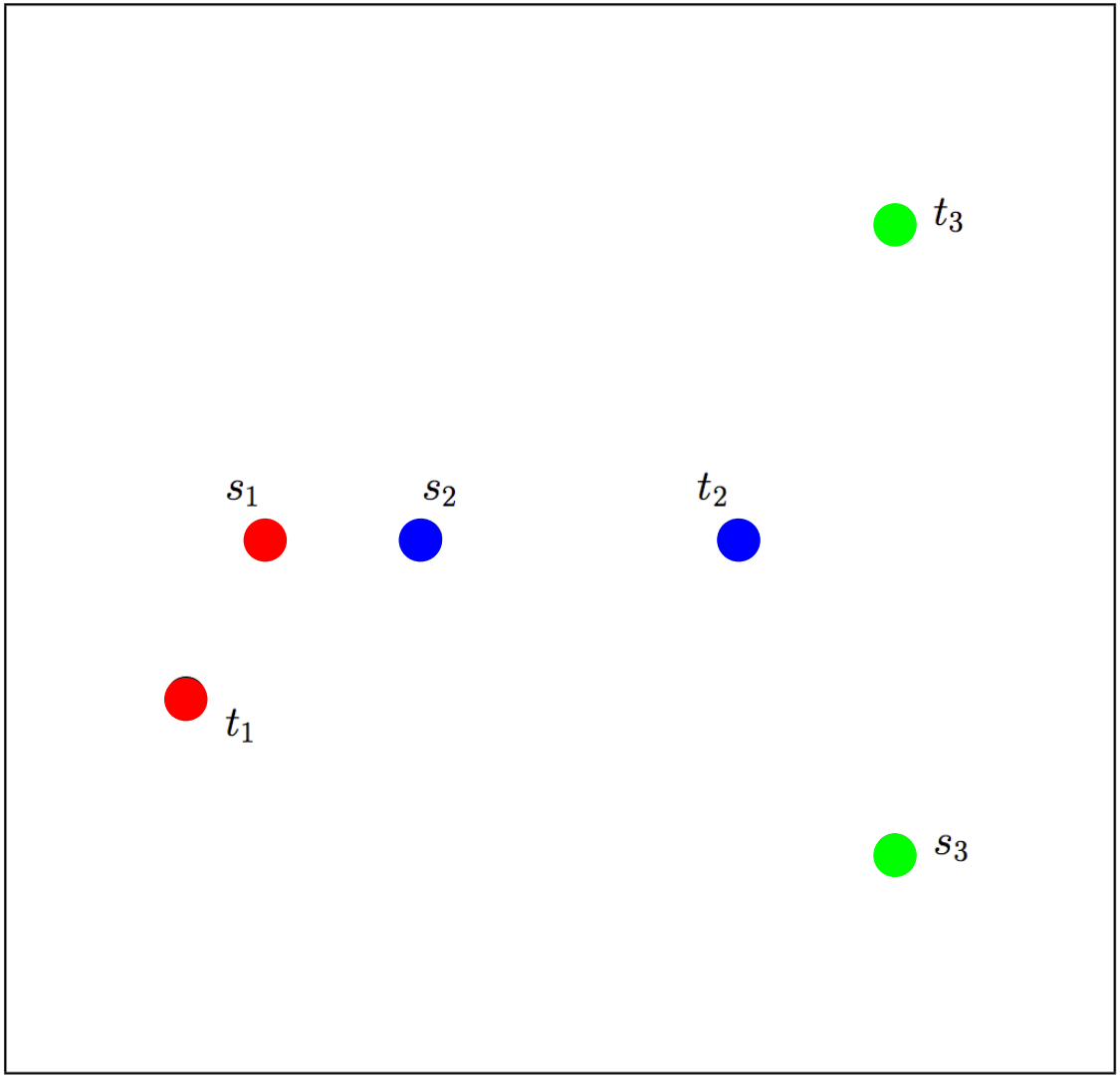
# Primal-Dual Algorithm

## Algorithm 1

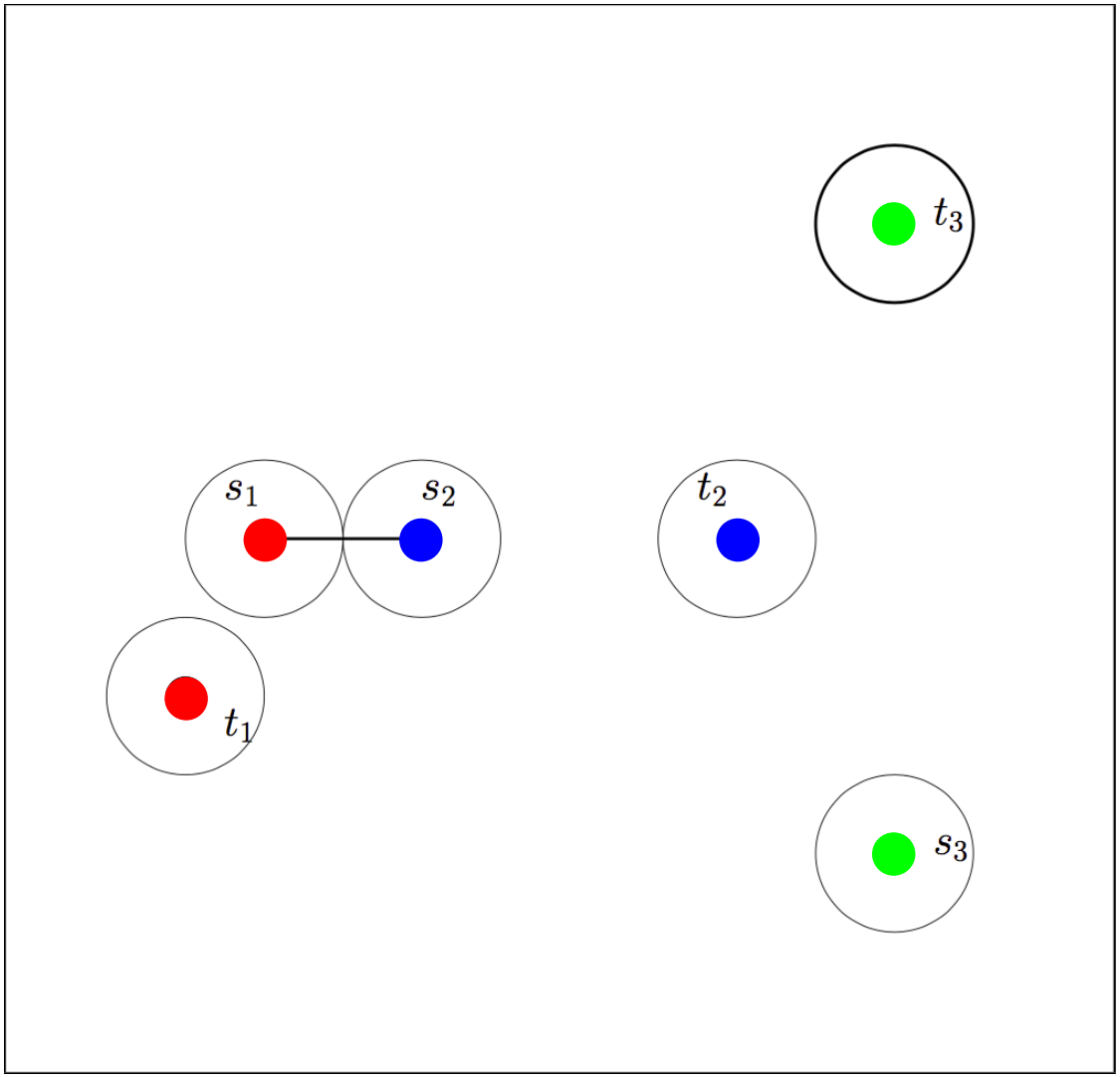
Initialize $F = \emptyset, y = 0$.

1. **while** not all $(s_i, t_i)$ pairs are connected in $(V, F)$ **do**
   - (a) Compute the set of all connected components $C$ of $(V, F)$ that separate terminal pairs.
   - (b) Increase $y_C$ uniformly for all of these sets $C$ until for some $e \in E, c_e = \sum_{S:e\in\delta(S)} y_S$
   - (c) $F \leftarrow F \cup \{e\}$
2. **return** $F$

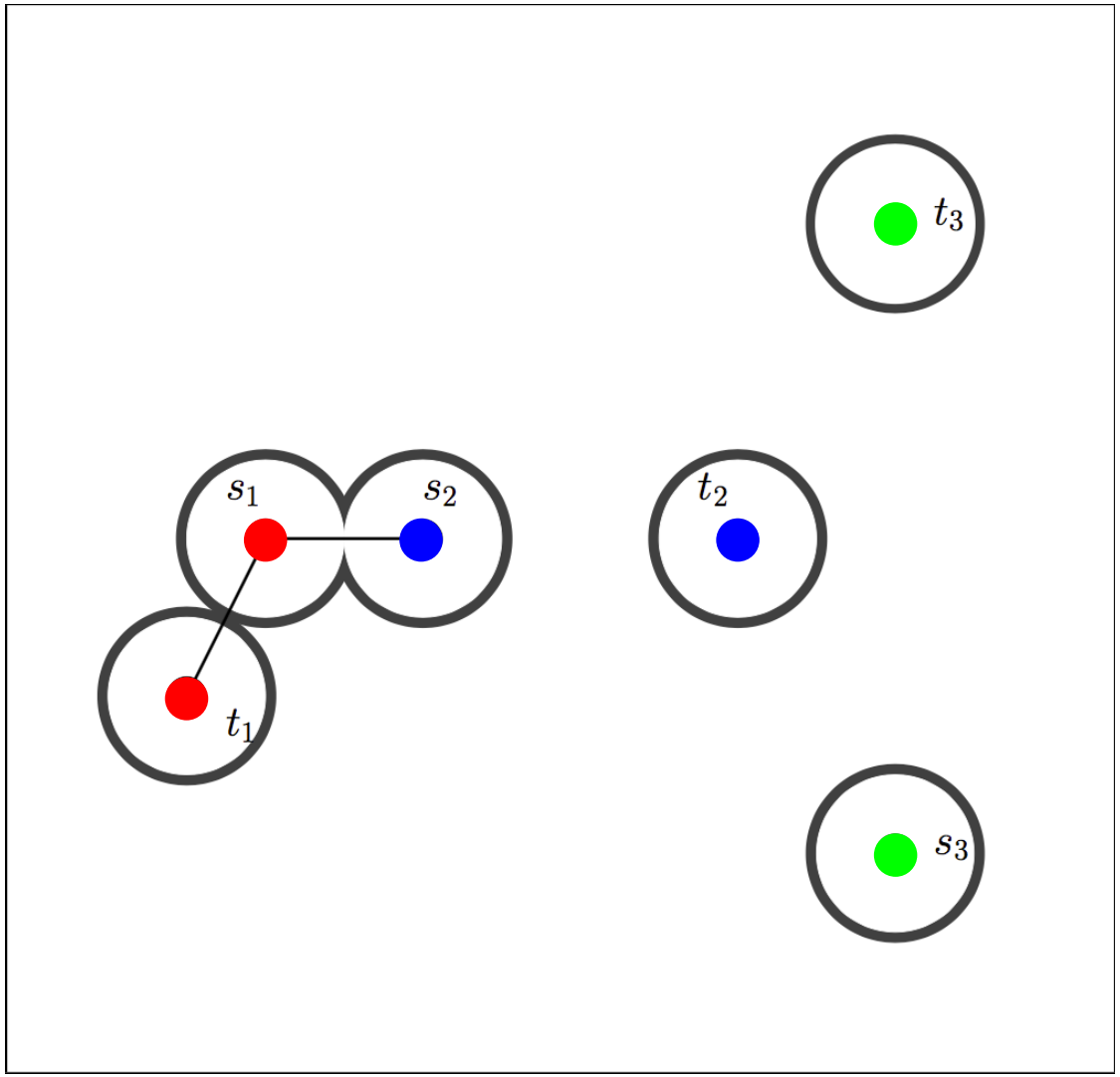
# Illustration of Algorithm



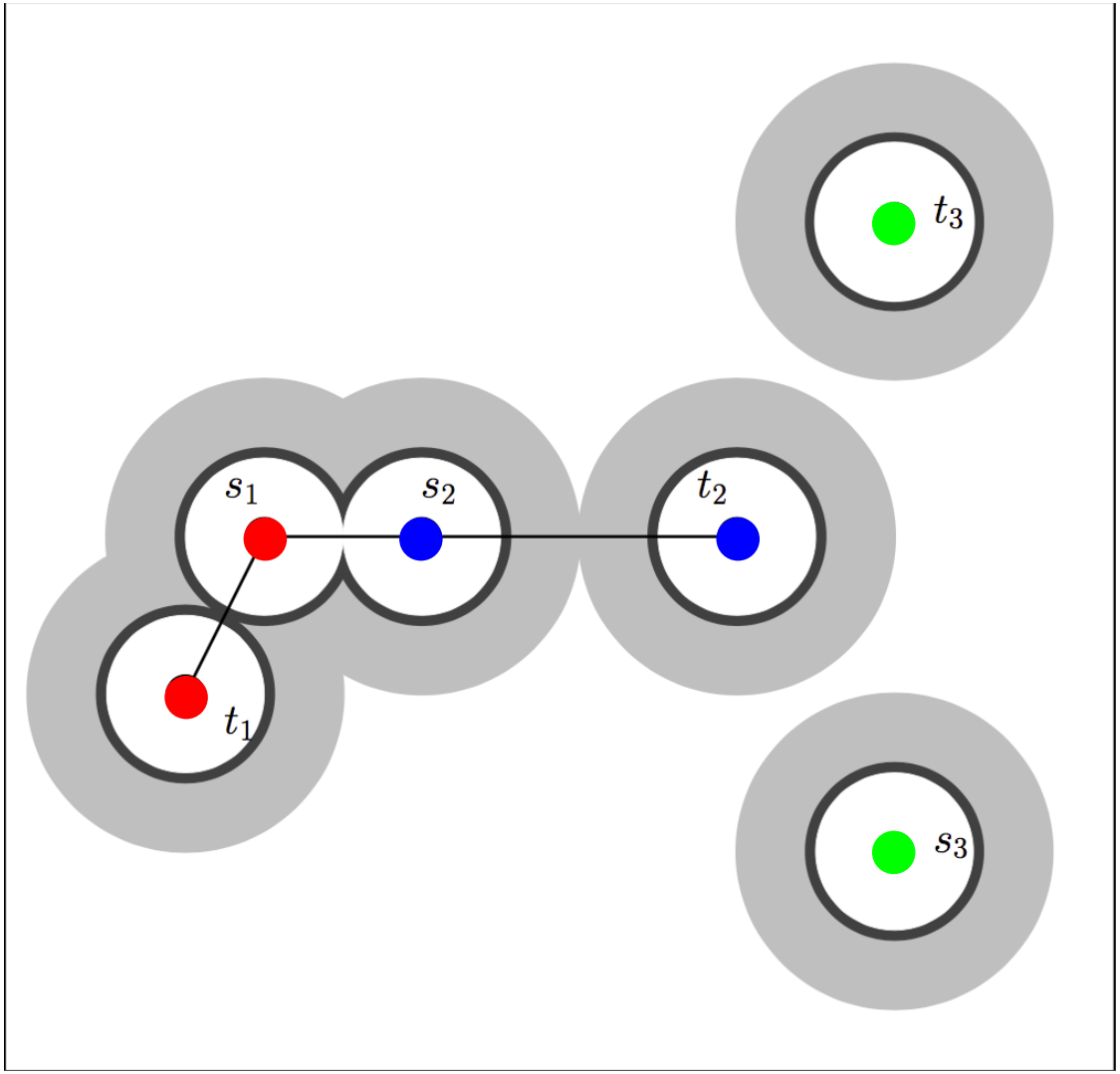
# Illustration of Algorithm



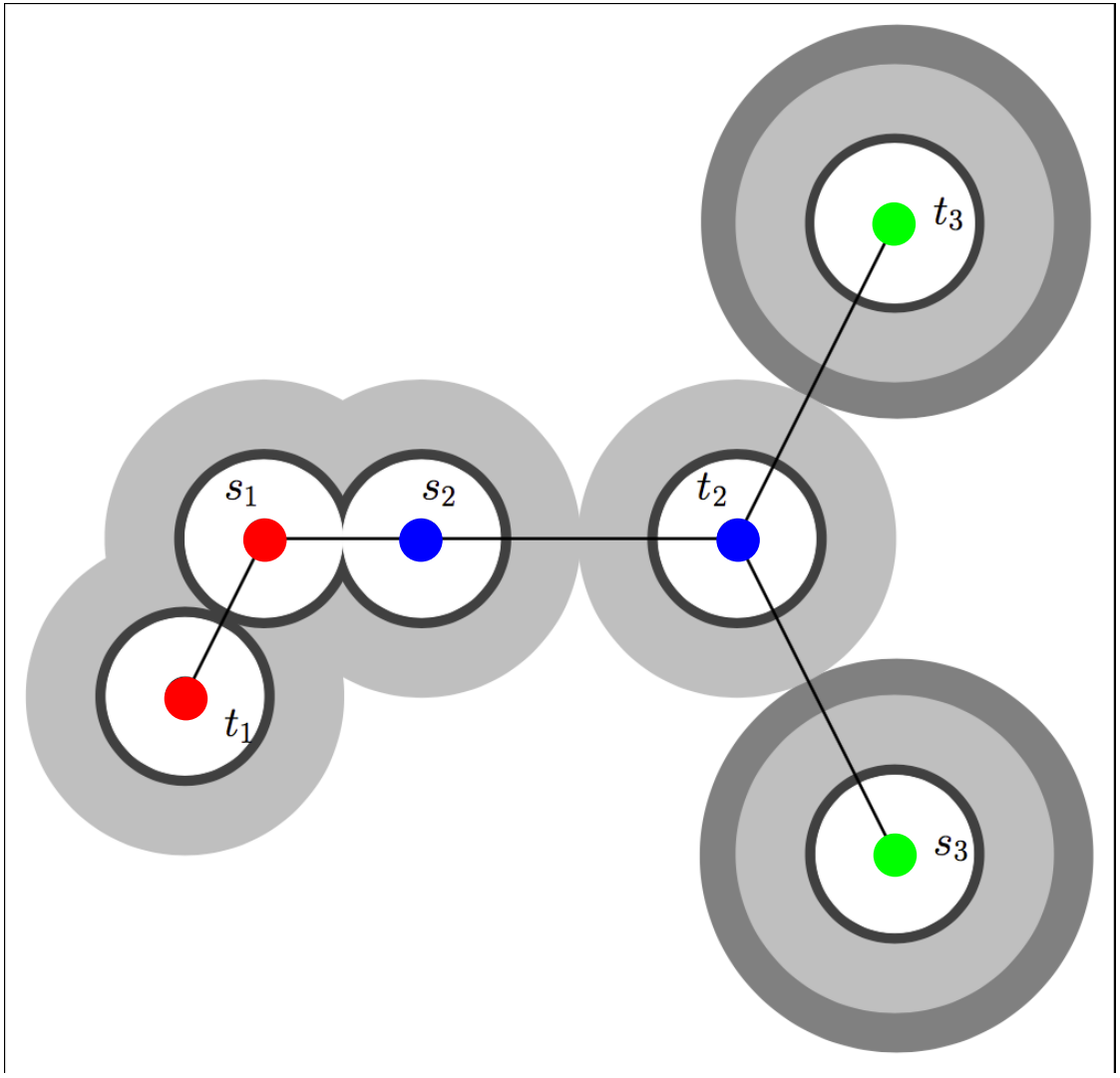
# Illustration of Algorithm



# Illustration of Algorithm



# Illustration of Algorithm



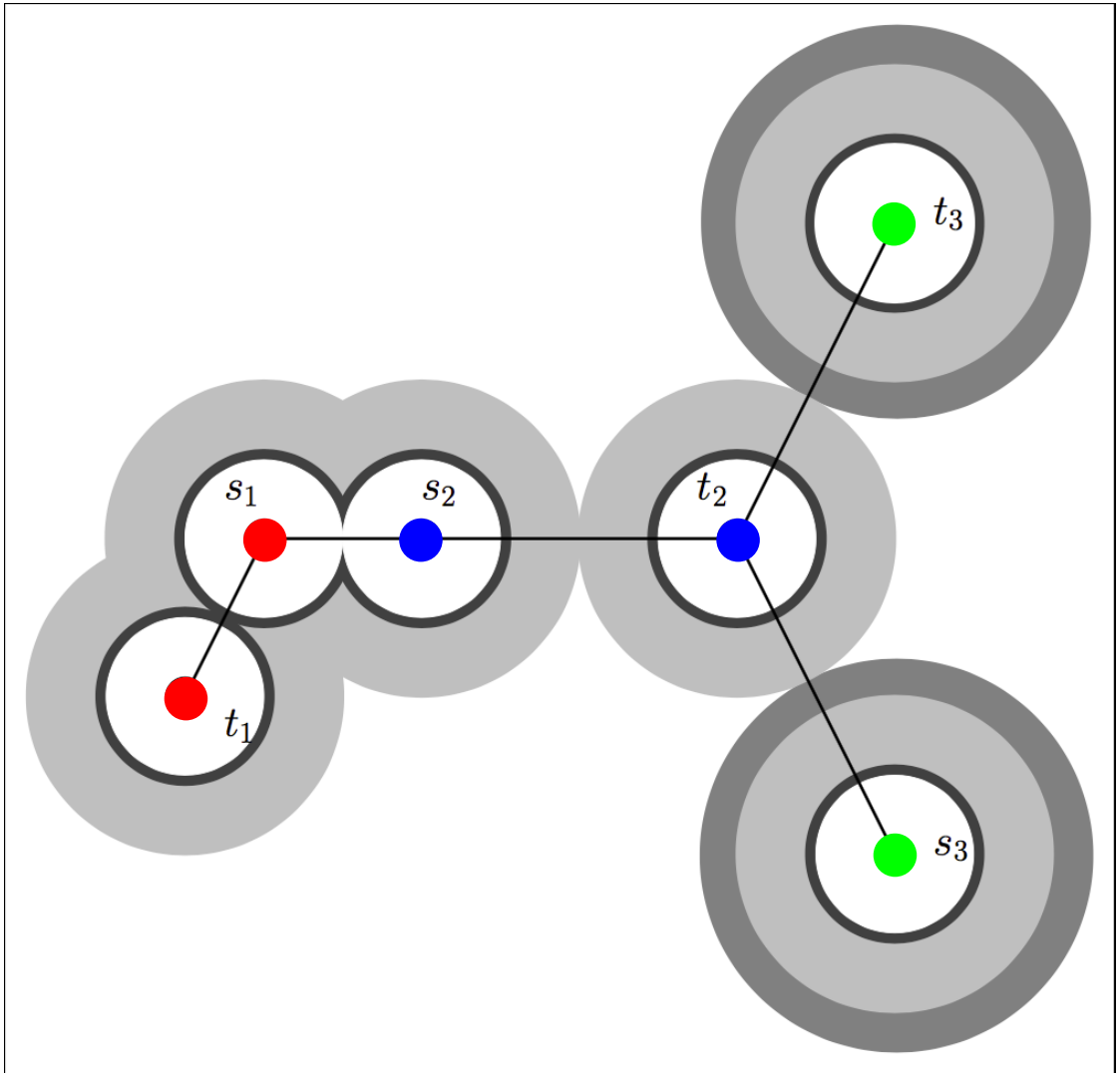
# Reverse Delete

## Algorithm 1 revised

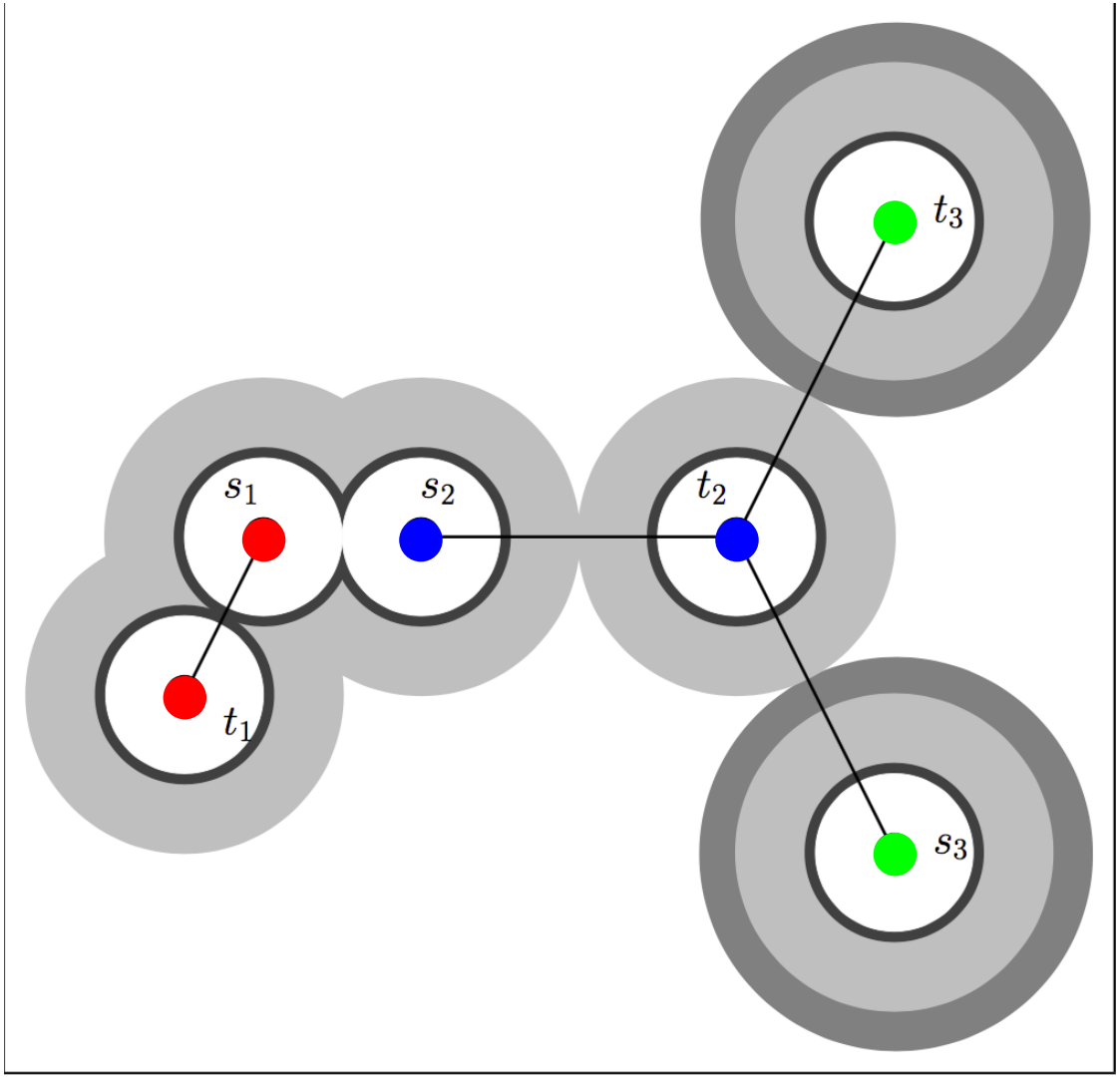
Initialize $F = \emptyset, y = 0, \ell = 0$.

1. $\ell \leftarrow \ell + 1$
2. **while** not all $(s_i, t_i)$ pairs are connected in $(V, F)$ **do**
   (a) Compute the set of all connected components $C$ of $(V, F)$ that are separate terminal pairs
   (b) Increase $y_C$ uniformly for all of these sets $C$ until for some $e_\ell \in E, c_{e_\ell} = \sum_{S: e_\ell \in \delta(S)} y_S$
   (c) $F \leftarrow F \cup \{e_\ell\}$
3. **for** $k \leftarrow \ell$ down to 1 **do**
   (a) **if** $F - e_k$ is a feasible solution **then** remove $e_k$ from $F$
4. **return** $F$

# Illustration of Algorithm



# Illustration of Algorithm



# Approximation Analysis

Note that every edge $e \in F$ corresponds to a tight constraint in $(D)$. So the total cost of our solution is

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

# Approximation Analysis

Note that every edge $e \in F$ corresponds to a tight constraint in $(D)$. So the total cost of our solution is

$$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S : e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| y_S.$$

To get a 2-approximation, it suffices to show

$$\sum_S |\delta(S) \cap F| y_S \leq 2 \sum_S y_S$$

since then by duality and the relaxation of the LP we have

$$\sum_{e \in F} c_e \leq 2 \sum_S y_S \leq 2 \cdot \mathrm{OPT}$$

# Technical Lemma

## Lemma

For any iteration of the algorithm, we have

$$\sum_{C} |\delta(C) \cap F| \leq 2 \cdot (\#\text{ of sets } C).$$

In our example, this means the number of times that the edges in the solution cross a moat is at most twice the number of moats.

# Proof of 2-approximation

**Theorem**

Algorithm 1 is a 2-approximation algorithm.

*Proof.* Recall that it suffices to show

$$\sum_{e \in F} c_e = \sum_S |\delta(S) \cap F| y_S \leq 2 \sum_S y_S \leq 2 \cdot OPT.$$

# Proof of 2-approximation

**Theorem**

Algorithm 1 is a 2-approximation algorithm.

*Proof.* Recall that it suffices to show

$$\sum_{e \in F} c_e = \sum_S |\delta(S) \cap F| y_S \leq 2 \sum_S y_S \leq 2 \cdot OPT.$$

We proceed by induction on the iterations of the algorithm. Initially all $y_S = 0$, so inequality holds. Assume that the inequality holds at the beginning of an iteration of the algorithm. In this iteration, we increase each dual variable $y_C$ by $\epsilon$. The increase in the L.H.S. is then $\epsilon \cdot \sum_C |\delta(C) \cap F|$ and the increase in the R.H.S. is $2\epsilon \cdot (\#$ of sets $C)$. But the technical lemma precisely says that $\sum_C |\delta(C) \cap F| \leq 2 \cdot (\#$ of sets $C)$. Therefore the increase in the L.H.S. is less than the increase in the R.H.S., and the result follows. □

# Table of Contents

# Research

- The above algorithm and related research was developed prior to 1995.

# Research

- The above algorithm and related research was developed prior to 1995.
- Network design problems have been further generalized in the last 20 years, and many new variants and special cases have been explored.

# Research: VC-SNDP

- There is current research on the Vertex Connectivity Survivable Network Design Problem (VC-SNDP).

# Research: VC-SNDP

- There is current research on the Vertex Connectivity Survivable Network Design Problem (VC-SNDP).
- We require a subgraph containing a specified number $r_{ij}$ of vertex-disjoint paths between $v_i$ and $v_j$.

# Research: VC-SNDP

- There is current research on the Vertex Connectivity Survivable Network Design Problem (VC-SNDP).
- We require a subgraph containing a specified number $r_{ij}$ of vertex-disjoint paths between $v_i$ and $v_j$.
- The generalized Steiner tree problem is VC-SNDP with all $r_{ij} \in \{0, 1\}$.

# Example

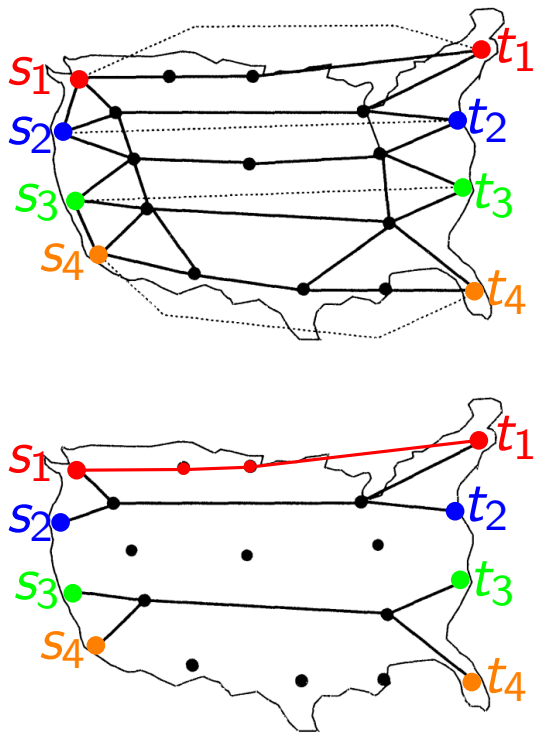



Figure: If $r_{s_1 t_1} = 2$ and other requirements are 1, then additional edges are required.

# Known Results

- VC-SNDP with requirements $r_{ij} \in \{0, 1, 2\}$ has a 2-approximation algorithm. The presented primal-dual algorithm does not work as is; instead, an augmentation framework is used.

# Known Results

- VC-SNDP with requirements $r_{ij} \in \{0, 1, 2\}$ has a 2-approximation algorithm. The presented primal-dual algorithm does not work as is; instead, an augmentation framework is used.
- Some special cases of VC-SNDP have algorithms with approximation factors independent of the number of terminals.

# Open Problems

Our interest this term:

- Does VC-SNDP with general requirements admit an approximation ratio independent of the number of terminals?

# Open Problems

Our interest this term:

- Does VC-SNDP with general requirements admit an approximation ratio independent of the number of terminals?
- Does VC-SNDP with $r_{ij} \in \{0, 1, 2, 3\}$ admit a constant ratio approximation?

# Open Problems

Our interest this term:

- Does VC-SNDP with general requirements admit an approximation ratio independent of the number of terminals?
- Does VC-SNDP with $r_{ij} \in \{0, 1, 2, 3\}$ admit a constant ratio approximation?

We are approaching these problems with a similar primal-dual technique as presented here, as well as other techniques.

# Open Problems

Our interest this term:

- Does VC-SNDP with general requirements admit an approximation ratio independent of the number of terminals?
- Does VC-SNDP with $r_{ij} \in \{0, 1, 2, 3\}$ admit a constant ratio approximation?

We are approaching these problems with a similar primal-dual technique as presented here, as well as other techniques.

Network design research still has many open problems and room for algorithmic improvements.

# Thank You!

Thank You!