Packing Steiner Trees

Mohammad R. Salavatipour

from two joint works with

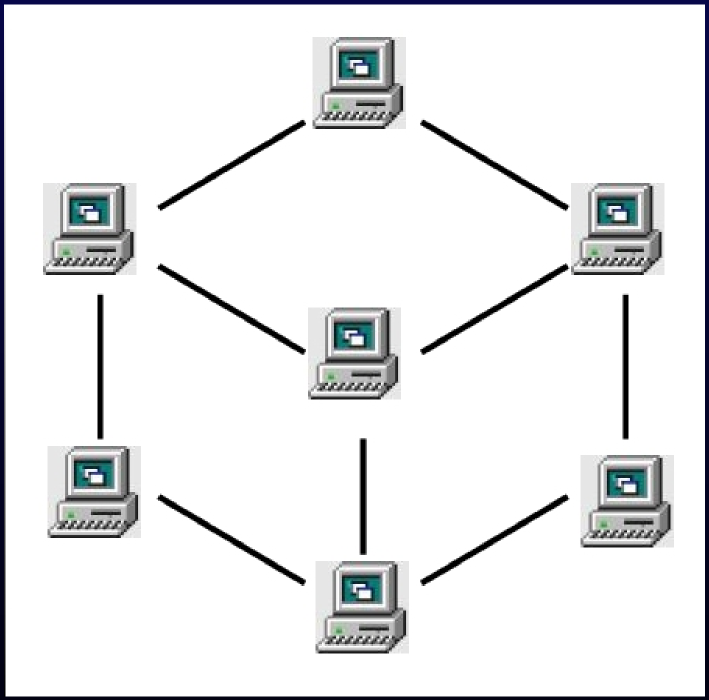
K. Jain (Microsoft) and M. Mahdian (MIT)

and

J. Cheriyan (Waterloo)

A Network Problem

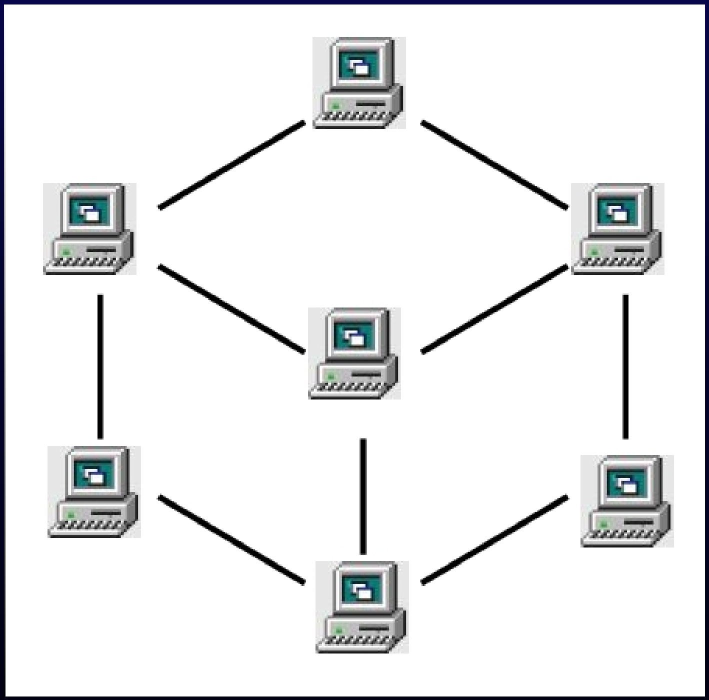
A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users



A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

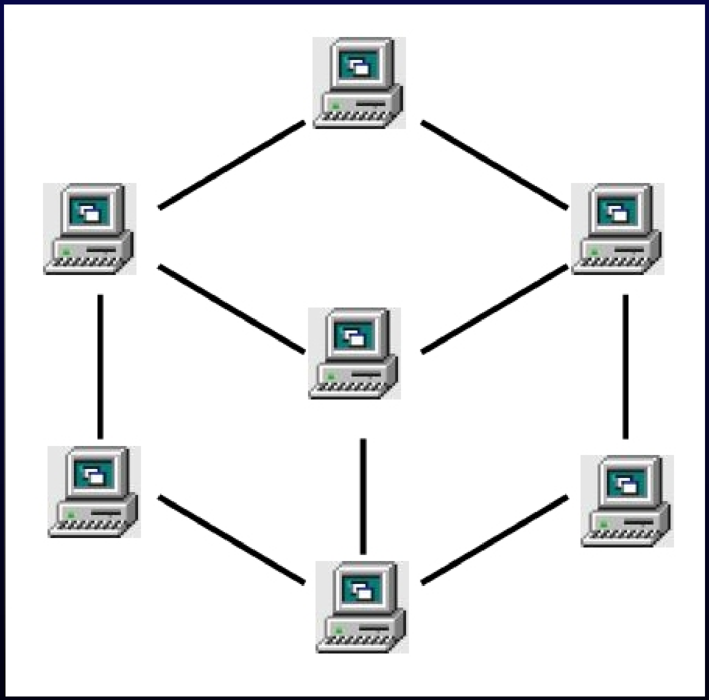
- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,



A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

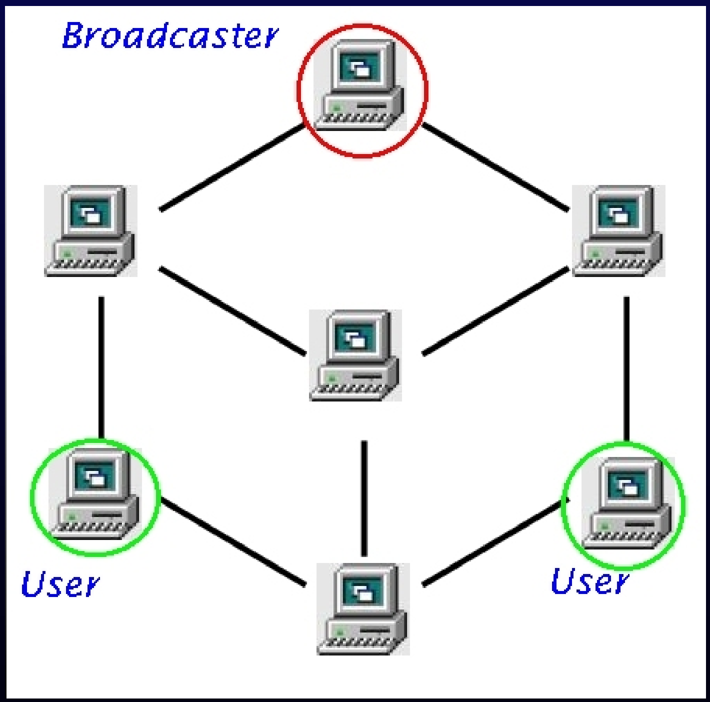
- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,



A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,

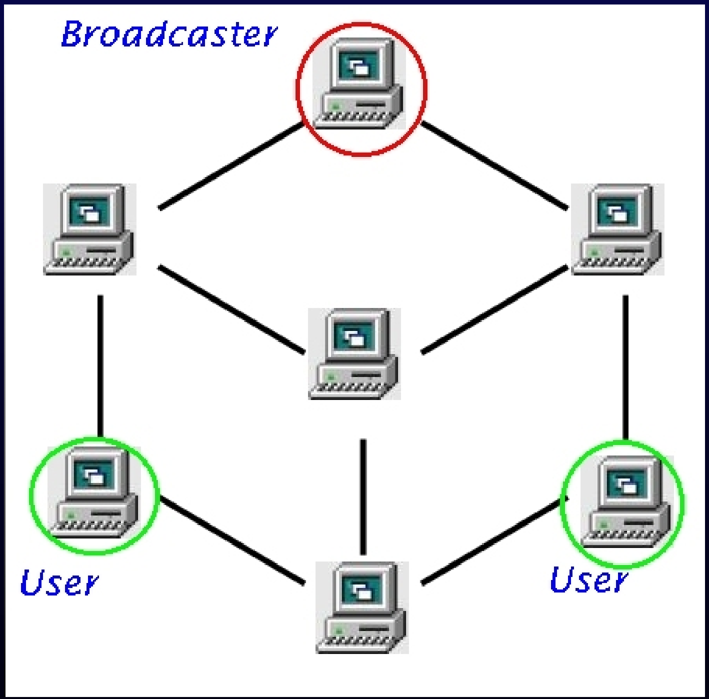


A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,

Each stream of video traverses a tree in N, rooted at the broadcaster, called Steiner tree.

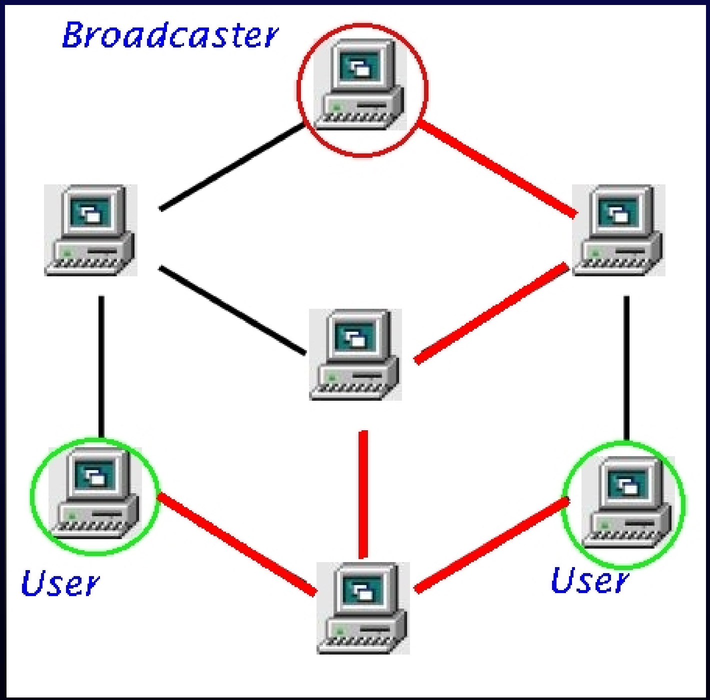


A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,

Each stream of video traverses a tree in N, rooted at the broadcaster, called Steiner tree.



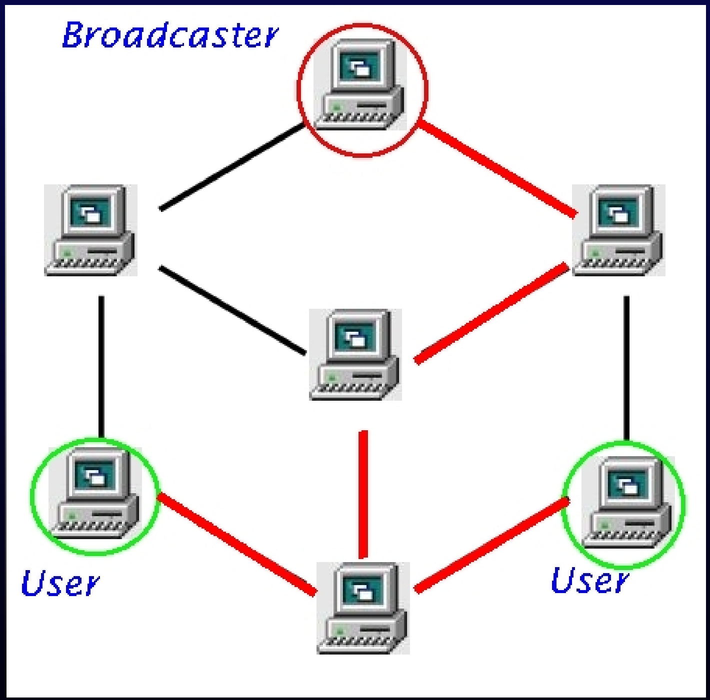
A Network Problem

A network N, a special node, called *broadcaster*, and we want to broadcast some streams of video to some users

- *Users (terminals)*: Are those nodes which have requested these streams,
- *Routers*: All nodes can pass the data,

Each stream of video traverses a tree in N, rooted at the broadcaster, called Steiner tree.

Goal: Find maximum number of edge-disjoint Steiner trees.



To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2$

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2 \implies$ Steiner trees are basically paths between two nodes

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2 \implies$ Steiner trees are basically paths between two nodes $\implies$ Theorem (Menger 1920's): The number of edge-disjoint paths between two vertices u and v is equal to the minimum number of edges whose removal disconnects u and v, and we can easily find the solution in linear time.

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2 \implies$ Steiner trees are basically paths between two nodes $\implies$ Theorem (Menger 1920's): The number of edge-disjoint paths between two vertices u and v is equal to the minimum number of edges whose removal disconnects u and v, and we can easily find the solution in linear time.

If $S = V(G)$

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2 \implies$ Steiner trees are basically paths between two nodes $\implies$ Theorem (Menger 1920's): The number of edge-disjoint paths between two vertices u and v is equal to the minimum number of edges whose removal disconnects u and v, and we can easily find the solution in linear time.

If $S = V(G) \implies$ Steiner trees are spanning trees

To Graph Theory

Given a graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes.

Find maximum number of edge-disjoint Steiner trees in G.

The two extreme cases of the problem are fundamental theorems:

If $|T| = 2 \implies$ Steiner trees are basically paths between two nodes $\implies$
Theorem (Menger 1920's): The number of edge-disjoint paths between two vertices u and v is equal to the minimum number of edges whose removal disconnects u and v, and we can easily find the solution in linear time.

If $S = V(G) \implies$ Steiner trees are spanning trees $\implies$
Theorem (Nash-Williams & Tutte 1960's): G has k edge-disjoint spanning trees iff for every partition $\mathcal{P} = \{V_1, \ldots, V_p\}$ of V:

$$E_G(\mathcal{P}) \geq k(p-1),$$

where $E_G(\mathcal{P})$ is the number of edges between classes of $\mathcal{P}$.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.(Open for $k \geq 2$!)

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.(Open for $k \geq 2$!)

Theorem [Petingi, Rodriguez'00]: If G has T-edge-connectivity at least $2(3/2)^{|V(G)-T|}.k$, then there are k edge-disjoint Steiner trees in G.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.(Open for $k \geq 2$!)

Theorem [Petingi, Rodriguez'00]: If G has T-edge-connectivity at least $2(3/2)^{|V(G)-T|}.k$, then there are k edge-disjoint Steiner trees in G.

Theorem [Frank, Király, Kriesell'01]: If $G-T$ is independent set and the T-edge-connectivity of G is $3k$, then there are k edge-disjoint Steiner trees in G.

Some earlier results

For $T \subseteq V$, *T-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.(Open for $k \geq 2$!)

Theorem [Petingi, Rodriguez'00]: If G has T-edge-connectivity at least $2(3/2)^{|V(G)-T|}.k$, then there are k edge-disjoint Steiner trees in G.

Theorem [Frank, Király, Kriesell'01]: If $G-T$ is independent set and the T-edge-connectivity of G is $3k$, then there are k edge-disjoint Steiner trees in G.This also gives a polynomial. time algorithm.

Some earlier results

For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Corollary: If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

Conjecture [Kriesell'99]: If G has T-edge-connectivity at least $2k$, then there are at least k edge-disjoint Steiner trees in G.(Open for $k \geq 2$!)

Theorem [Petingi, Rodriguez'00]: If G has T-edge-connectivity at least $2(3/2)^{|V(G)-T|}.k$, then there are k edge-disjoint Steiner trees in G.

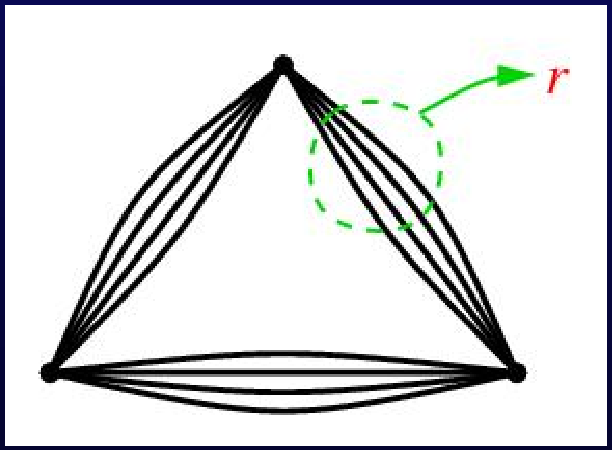
Theorem [Frank, Király, Kriesell'01]: If $G - T$ is independent set and the T-edge-connectivity of G is $3k$, then there are k edge-disjoint Steiner trees in G.This also gives a polynomial. time algorithm.

Theorem 1 (Jain & Mahdian & S.'03): If $|T| = t$ and G has T-edge-connectivity at least $(\frac{t}{4} + o(t))k$, then we can find k edge-disjoint Steiner trees in poly. time.

Theorem 2 (Jain & Mahdian & S. ’03): If $|T| = 3$ and G has T-edge-connectivity at least $\frac{4}{3}k$, then we can find k edge-disjoint Steiner trees in G.

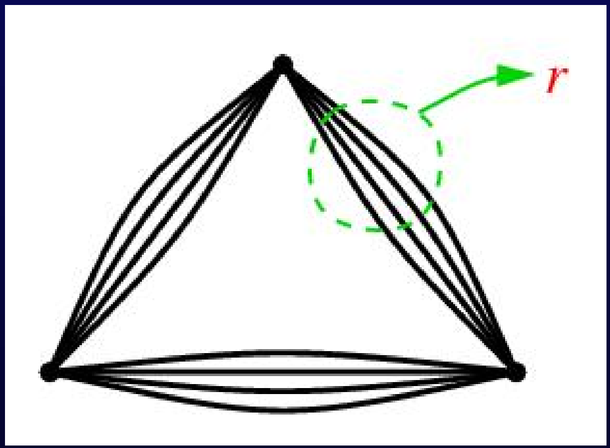
Theorem 2 (Jain & Mahdian & S. '03): If $|T| = 3$ and G has T-edge-connectivity at least $\frac{4}{3}k$, then we can find k edge-disjoint Steiner trees in G.

Example showing tightness: For this graph $T = V$ and it is $2r$-edge-connected; number of Steiner trees is $\frac{3r}{2} = \frac{3}{4} \times 2r$.



Theorem 2 (Jain & Mahdian & S. ’03): If $|T| = 3$ and G has T-edge-connectivity at least $\frac{4}{3}k$, then we can find k edge-disjoint Steiner trees in G.

Example showing tightness: For this graph $T = V$ and it is $2r$-edge-connected; number of Steiner trees is $\frac{3r}{2} = \frac{3}{4} \times 2r$.



Theorem (Lau’04, unpublished): If G has T-edge-connectivity at least $26k$, then we can find k edge-disjoint Steiner trees in poly. time.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Theorem 4 (Cheriyan & S.): There is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$, (i.e. it is APX-hard).

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Theorem 4 (Cheriyan & S.): There is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$, (i.e. it is APX-hard).

Proof idea: A reduction from Bounded 3-Dimensional-Matching (B3DM).

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Theorem 4 (Cheriyan & S.): There is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$, (i.e. it is APX-hard).

Proof idea: A reduction from Bounded 3-Dimensional-Matching (B3DM). Given instance G of B3DM with m edges construct H with 4 terminals s.t.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Theorem 4 (Cheriyan & S.): There is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$, (i.e. it is APX-hard).

Proof idea: A reduction from Bounded 3-Dimensional-Matching (B3DM). Given instance G of B3DM with m edges construct H with 4 terminals s.t.

- if G has a perfect matching then H has m Steiner trees.

Algorithmic Point of View

Packing Steiner trees is also interesting from the algorithmic point of view.

Let PEU denote the problem of finding maximum number of Edge-disjoint Undirected Steiner trees.

Not surprisingly, the problem is NP-complete:
Theorem 3 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees.

How about when $|T|$ is constant?

Theorem 4 (Cheriyan & S.): There is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$, (i.e. it is APX-hard).

Proof idea: A reduction from Bounded 3-Dimensional-Matching (B3DM).
Given instance G of B3DM with m edges construct H with 4 terminals s.t.

- if G has a perfect matching then H has m Steiner trees.
- if max matching of G is $\leq (1-\epsilon)m$ then H has at most $(1-\frac{\epsilon}{100})m$ trees.

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T : e \in T} x_T \le c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0, 1\} \end{array}$$

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T : e \in T} x_T \leq c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0, 1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T : e \in T} x_T \leq c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0, 1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e y_e \\ \text{subject to} & \forall T \in \mathcal{T} : \sum_{e \in T} y_e \geq 1 \\ & \forall e \in E : \quad y_e \geq 0 \end{array}$$

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T : e \in T} x_T \leq c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0, 1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e y_e \\ \text{subject to} & \forall T \in \mathcal{T} : \sum_{e \in T} y_e \geq 1 \\ & \forall e \in E : \quad y_e \geq 0 \end{array}$$

The separation oracle for the dual LP is the min. Steiner Tree problem:

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T : e \in T} x_T \leq c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0, 1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e y_e \\ \text{subject to} & \forall T \in \mathcal{T} : \sum_{e \in T} y_e \geq 1 \\ & \forall e \in E : \quad y_e \geq 0 \end{array}$$

The separation oracle for the dual LP is the min. Steiner Tree problem:

Given weighted graph G and set T find a min weight Steiner tree.

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T\in\mathcal{T}} x_T \\ \text{subject to} & \forall e\in E: \sum_{T:e\in T} x_T \le c_e \\ & \forall T\in\mathcal{T}: \quad x_T \in \{0,1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

$$\begin{array}{ll} \text{minimize} & \sum_{e\in E} c_e y_e \\ \text{subject to} & \forall T\in\mathcal{T}: \sum_{e\in T} y_e \ge 1 \\ & \forall e\in E: \quad y_e \ge 0 \end{array}$$

The separation oracle for the dual LP is the min. Steiner Tree problem:

Given weighted graph G and set T find a min weight Steiner tree.

Theorem 5 (Jain & Mahdian & S.'03): There is an α-approx algorithm for fractional PEU iff there is an α-approx algorithm for min Steiner tree.

LP formulation and fractional packing

We can formulate PEU as an ILP. Let $\mathcal{T}$ be the set of all Steiner trees.

$$\begin{array}{ll} \text{maximize} & \sum_{T\in\mathcal{T}} x_T \\ \text{subject to} & \forall e \in E : \sum_{T: e\in T} x_T \leq c_e \\ & \forall T \in \mathcal{T} : \quad x_T \in \{0,1\} \end{array}$$

Fractional PEU is the corresponding LP. The dual LP will be:

$$\begin{array}{ll} \text{minimize} & \sum_{e\in E} c_e y_e \\ \text{subject to} & \forall T \in \mathcal{T} : \sum_{e\in T} y_e \geq 1 \\ & \forall e \in E : \quad y_e \geq 0 \end{array}$$

The separation oracle for the dual LP is the min. Steiner Tree problem:

Given weighted graph G and set T find a min weight Steiner tree.

Theorem 5 (Jain & Mahdian & S.'03): There is an α-approx algorithm for fractional PEU iff there is an α-approx algorithm for min Steiner tree.

Corollary: Fractional PEU is APX-hard and has an 1.59-approx algorithm.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 7 (Cheriyan & S.): PVU is APX-hard even if $|T| = 4$.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 7 (Cheriyan & S.): PVU is APX-hard even if $|T| = 4$.

Will come back to PVU at the end of the talk.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 7 (Cheriyan & S.): PVU is APX-hard even if $|T| = 4$.

Will come back to PVU at the end of the talk.

We can also define the same problems in the directed version.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 7 (Cheriyan & S.): PVU is APX-hard even if $|T| = 4$.

Will come back to PVU at the end of the talk.

We can also define the same problems in the directed version.

Packing Edge-disjoint Direct Steiner trees (PED): Given directed graph G and terminals $T \subseteq V$ containing a root r, find max number of edge-disjoint (rooted) Steiner trees.

Other variations

Packing Vertex-disjoint Undirected Steiner trees (PVU): Given undirected graph G and terminals $T \subseteq V$, find max number of Steiner trees that are *internally* vertex disjoint (i.e. on Steiner nodes).

The same results we proved for PEU also hold for PVU:

Theorem 6 (Cheriyan & S.): Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 7 (Cheriyan & S.): PVU is APX-hard even if $|T| = 4$.

Will come back to PVU at the end of the talk.

We can also define the same problems in the directed version.

Packing Edge-disjoint Direct Steiner trees (PED): Given directed graph G and terminals $T \subseteq V$ containing a root r, find max number of edge-disjoint (rooted) Steiner trees.

Packing Vertex-disjoint Direct Steiner trees (PVD): Similar to PED, except that trees have to be disjoint on Steiner nodes.

By easy reductions, we can show that PED and PVD are equally hard:

By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

Proof: Basic idea is let G' be the line graph of G.

By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

Proof: Basic idea is let G' be the line graph of G.

G' contains all the terminals of G as terminals (and root as root),

By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

Proof: Basic idea is let G' be the line graph of G.

G' contains all the terminals of G as terminals (and root as root), and

one Steiner node v_{xy} for every
edge $xy \in E(G)$.

By easy reductions, we can show that PED and PVD are equally hard:

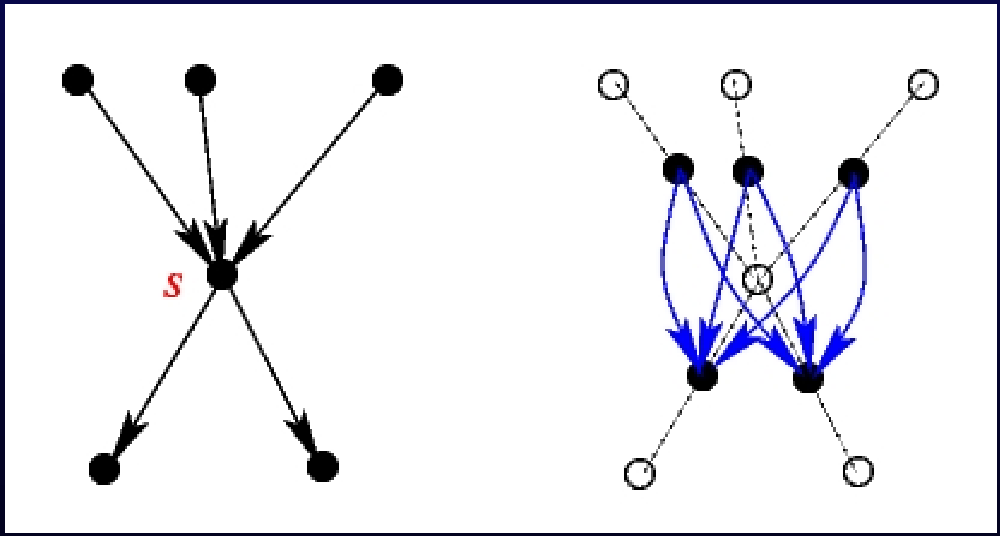
Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

Proof: Basic idea is let G' be the line graph of G.

G' contains all the terminals of G as terminals (and root as root), and

one Steiner node v_{xy} for every edge $xy \in E(G)$.

For each $s \in V(G)$ we add the following edge to G':



By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

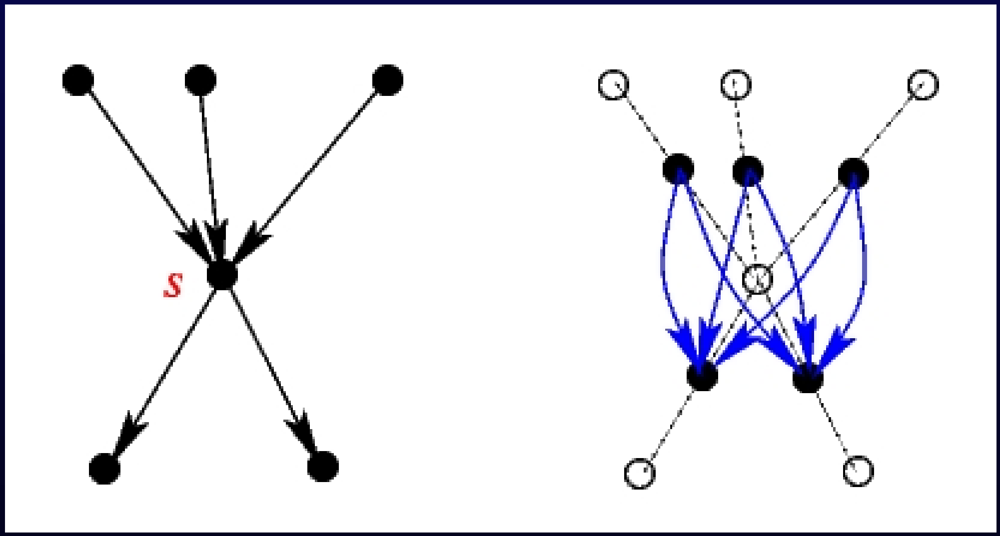
Proof: Basic idea is let G' be the line graph of G.

G' contains all the terminals of G as terminals (and root as root), and

one Steiner node v_{xy} for every edge $xy \in E(G)$.

For each $s \in V(G)$ we add the following edge to G':

edge-disjoint Steiner trees in G correspond to vertex-disjoint Steiner trees in G', and vice versa.



Similarly, we can reduce PVD to PED.

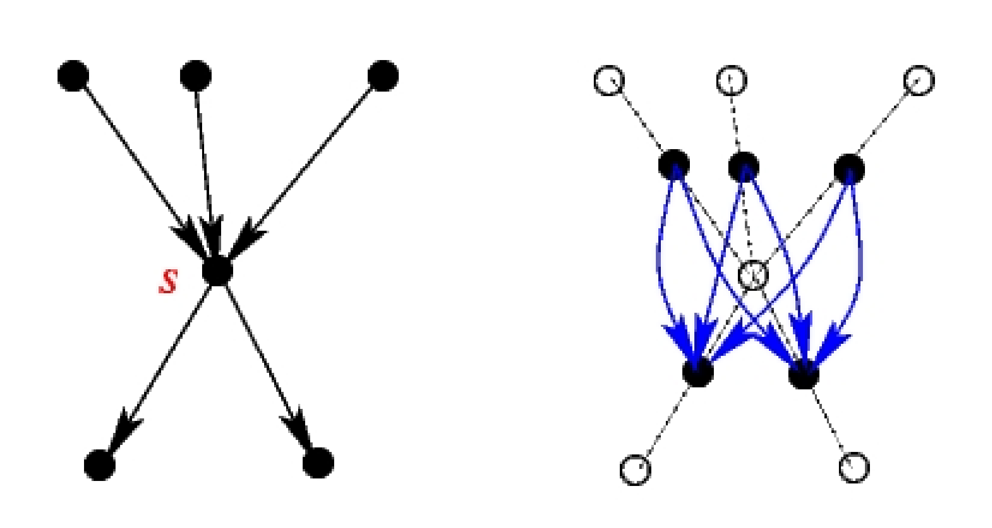
By easy reductions, we can show that PED and PVD are equally hard:

Theorem 8 (Cheriyan & S.): Given an instance $I = (G, k)$ of PED, there is an instance $I' = (G', k)$ of PVD with $|G'| = poly(|G|)$, such that G has k edge-disjoint directed Steiner trees iff G' has k vertex-disjoint Steiner trees.

Proof: Basic idea is let G' be the line graph of G.

G' contains all the terminals of G as terminals (and root as root), and

one Steiner node v_{xy} for every edge $xy \in E(G)$.



For each $s \in V(G)$ we add the following edge to G':

edge-disjoint Steiner trees in G correspond to vertex-disjoint Steiner trees in G', and vice versa.

Similarly, we can reduce PVD to PED.

Therefore, we only focus on finding algorithms and proving hardness for PED.

L. Lau showed that there is a 26-approx algorithm for PEU.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP

2. Relax it to LP (i.e. consider the fractional PED)

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)
4. use randomized rounding to obtain an integral solution.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)
4. use randomized rounding to obtain an integral solution.

Take the LP corresponding to Fractional PED and and consider the dual LP.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)
4. use randomized rounding to obtain an integral solution.

Take the LP corresponding to Fractional PED and and consider the dual LP.

The separation oracle for the dual is min directed Steiner tree problem.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)
4. use randomized rounding to obtain an integral solution.

Take the LP corresponding to Fractional PED and and consider the dual LP.

The separation oracle for the dual is min directed Steiner tree problem.

Min Directed Steiner: Given directed weighted graph G and $T \subseteq V$ containing a root r, find min weight (rooted) Steiner tree.

L. Lau showed that there is a 26-approx algorithm for PEU.
How about PED?

Theorem 9 (Cheriyan & S.): For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

The basic idea is:

1. Formulate PED as an ILP
2. Relax it to LP (i.e. consider the fractional PED)
3. Try to solve this LP (maybe approximately)
4. use randomized rounding to obtain an integral solution.

Take the LP corresponding to Fractional PED and and consider the dual LP.

The separation oracle for the dual is min directed Steiner tree problem.

Min Directed Steiner: Given directed weighted graph G and $T \subseteq V$ containing a root r, find min weight (rooted) Steiner tree.
This is NP-hard, even hard to approximate within $O(\log^2 n)$ factor.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^\epsilon)$, for any $\epsilon > 0$.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^{\epsilon})$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:

Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^{\epsilon})$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:

Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^{\epsilon})$-approx algorithm for fractional PED.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^{\epsilon})$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:
Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^{\epsilon})$-approx algorithm for fractional PED.

Main Lemma: If I is an instance of PED and φ is the value of a feasible solution to the fractional instance I_f, then we can find a solution to I with value at least $O(\frac{\varphi}{\sqrt{m}})$.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^\epsilon)$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:
Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^\epsilon)$-approx algorithm for fractional PED.

Main Lemma: If I is an instance of PED and φ is the value of a feasible solution to the fractional instance I_f, then we can find a solution to I with value at least $O(\frac{\varphi}{\sqrt{m}})$.

Proof: If $\varphi \leq \sqrt{m}$ simply find one Steiner tree and return it.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^\epsilon)$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:
Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^\epsilon)$-approx algorithm for fractional PED.

Main Lemma: If I is an instance of PED and φ is the value of a feasible solution to the fractional instance I_f, then we can find a solution to I with value at least $O(\frac{\varphi}{\sqrt{m}})$.

Proof: If $\varphi \leq \sqrt{m}$ simply find one Steiner tree and return it.

Else, for every tree $T \in \mathcal{T}$ with $x_T > 0$, pick it with prob. $x_T/\sqrt{m}$.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^\epsilon)$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:
Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^\epsilon)$-approx algorithm for fractional PED.

Main Lemma: If I is an instance of PED and φ is the value of a feasible solution to the fractional instance I_f, then we can find a solution to I with value at least $O(\frac{\varphi}{\sqrt{m}})$.

Proof: If $\varphi \leq \sqrt{m}$ simply find one Steiner tree and return it.

Else, for every tree $T \in \mathcal{T}$ with $x_T > 0$, pick it with prob. $x_T/\sqrt{m}$.

Define $X_T = 1$ iff we pick tree T, and let $X = \sum_{T \in \mathcal{T}} X_T$.

Theorem (Charikar, Chekuri, Cheung, Dai, Goel, Guha, & Li'99): Min. directed Steiner tree can be approximated within $O(n^\epsilon)$, for any $\epsilon > 0$.

Similar to Theorem 5, we can also prove:
Theorem 10: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

Corollary: There is an $O(n^\epsilon)$-approx algorithm for fractional PED.

Main Lemma: If I is an instance of PED and φ is the value of a feasible solution to the fractional instance I_f, then we can find a solution to I with value at least $O(\frac{\varphi}{\sqrt{m}})$.

Proof: If $\varphi \leq \sqrt{m}$ simply find one Steiner tree and return it.

Else, for every tree $T \in \mathcal{T}$ with $x_T > 0$, pick it with prob. $x_T/\sqrt{m}$.

Define $X_T = 1$ iff we pick tree T, and let $X = \sum_{T \in \mathcal{T}} X_T$.

$$\mathrm{E}[X] = \sum_{T \in \mathcal{T}} \Pr[X_T = 1] = \sum_{T \in \mathcal{T}} \frac{x_T}{\sqrt{m}} = \frac{\varphi}{\sqrt{m}}$$

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

It is easy to show that $\Pr[A_e] \leq \frac{10}{m}$.

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

It is easy to show that $\Pr[A_e] \leq \frac{10}{m}$. Thus:

$$\Pr[\bigwedge_{e \in E} \overline{A}_e] \geq \prod_{e \in E} \Pr[\overline{A}_e] \geq (1 - \frac{10}{m})^m \geq e^{-10}.$$

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

It is easy to show that $\Pr[A_e] \le \frac{10}{m}$. Thus:

$$\Pr[\bigwedge_{e \in E} \overline{A}_e] \ge \prod_{e \in E} \Pr[\overline{A}_e] \ge (1 - \frac{10}{m})^m \ge e^{-10}.$$

Also, by Chernoff bound, $\Pr[X < \frac{\mathrm{E}[X]}{10}] \le e^{-100\varphi/2\sqrt{m}} \le e^{-50}$

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

It is easy to show that $\Pr[A_e] \leq \frac{10}{m}$. Thus:

$$\Pr[\bigwedge_{e \in E} \overline{A}_e] \geq \prod_{e \in E} \Pr[\overline{A}_e] \geq (1 - \frac{10}{m})^m \geq e^{-10}.$$

Also, by Chernoff bound, $\Pr[X < \frac{\mathrm{E}[X]}{10}] \leq e^{-100\varphi/2\sqrt{m}} \leq e^{-50}$

Thus: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] < e^{-50} + 1 - e^{-10} < 1 - e^{-9}$,

For $e \in E$, let A_e be the bad event that two trees containing e are picked.

Goal: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] > 0$, i.e. there is an outcome of random trials s.t. the number of trees is at least $\mathrm{E}[X]/10 = \frac{v}{10\sqrt{m}}$ and no edge $e \in E$ belongs to more than 1 tree.

It is easy to show that $\Pr[A_e] \leq \frac{10}{m}$. Thus:

$$\Pr[\bigwedge_{e \in E} \overline{A}_e] \geq \prod_{e \in E} \Pr[\overline{A}_e] \geq (1 - \frac{10}{m})^m \geq e^{-10}.$$

Also, by Chernoff bound, $\Pr[X < \frac{\mathrm{E}[X]}{10}] \leq e^{-100\varphi/2\sqrt{m}} \leq e^{-50}$

Thus: $\Pr[(X < \frac{\mathrm{E}[X]}{10}) \vee (\exists e \in E : A_e)] < e^{-50} + 1 - e^{-10} < 1 - e^{-9}$,

So there is an outcome of X_T's, s.t. $(\bigwedge_{e \in E} \overline{A}_e) \wedge \left(X \geq \frac{\varphi}{10\sqrt{m}}\right)$. We can derandomize this using the method of conditional probabilities.

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!!

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

Remark: Proof is purely combinatorial and does not rely on PCP theorem.

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

Remark: Proof is purely combinatorial and does not rely on PCP theorem.

We use the following NP-hard problem, as the building block of our reduction:

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

Remark: Proof is purely combinatorial and does not rely on PCP theorem.

We use the following NP-hard problem, as the building block of our reduction:

PROBLEM: 2DIRPATH
INSTANCE: A directed graph $G(V, E)$, distinct vertices $x_1, y_1, x_2, y_2 \in V$.
QUESTION: Are there two edge-disjoint directed paths, one from x_1 to y_1 and the other from x_2 to y_2 in G?

There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

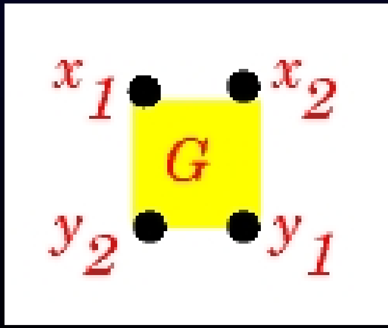
Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

Remark: Proof is purely combinatorial and does not rely on PCP theorem.

We use the following NP-hard problem, as the building block of our reduction:

PROBLEM: 2DIRPATH
INSTANCE: A directed graph $G(V, E)$, distinct vertices $x_1, y_1, x_2, y_2 \in V$.
QUESTION: Are there two edge-disjoint directed paths, one from x_1 to y_1 and the other from x_2 to y_2 in G?

Let $I = (G, x_1, y_1, x_2, y_2)$ be an instance of 2DIRPATH and $\epsilon > 0$ be given.



There is a huge gap between the approximation ratio for PEU (26) and PED ($O(m^{\frac{1}{2}+\epsilon})$)!! Is it just because the algorithm we gave is too dumb?

Theorem 11 (Cheriyan & S.): Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We prove the following weaker version here:

Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

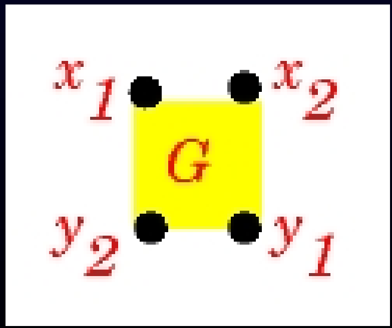
Remark: Proof is purely combinatorial and does not rely on PCP theorem.

We use the following NP-hard problem, as the building block of our reduction:

PROBLEM: 2DIRPATH
INSTANCE: A directed graph $G(V, E)$, distinct vertices $x_1, y_1, x_2, y_2 \in V$.
QUESTION: Are there two edge-disjoint directed paths, one from x_1 to y_1 and the other from x_2 to y_2 in G?

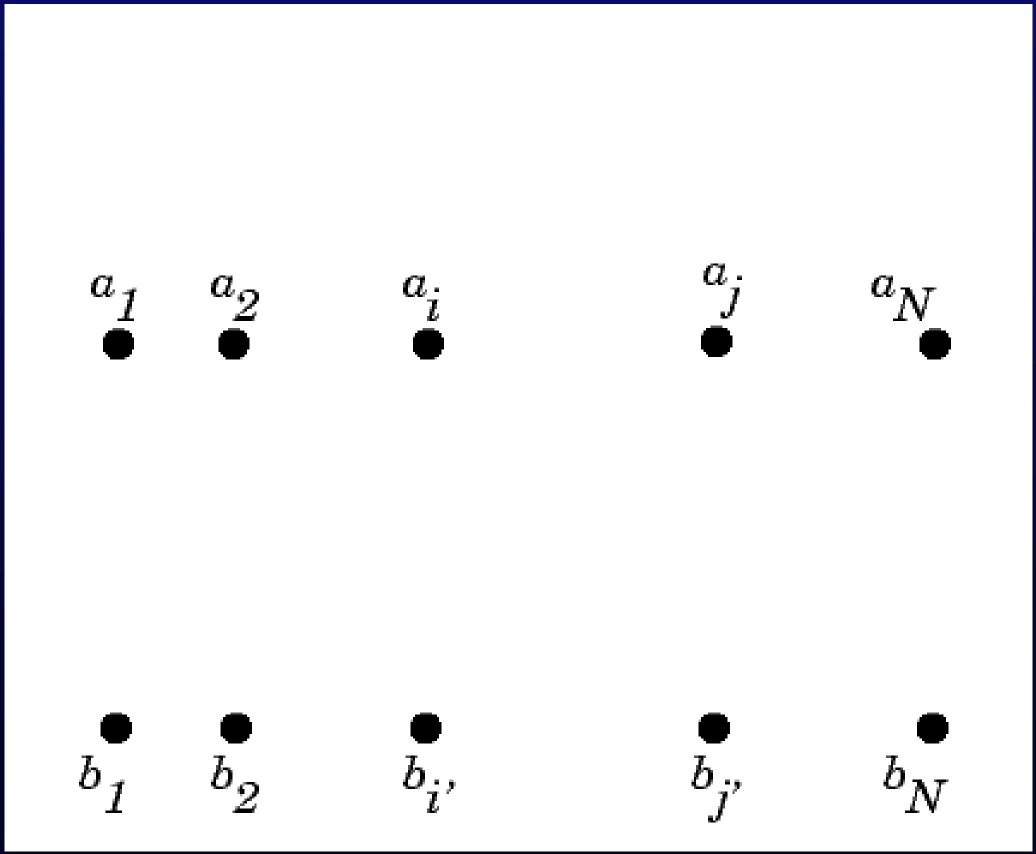
Let $I = (G, x_1, y_1, x_2, y_2)$ be an instance of 2DIRPATH and $\epsilon > 0$ be given.

We construct a digraph H which has several copies of G.



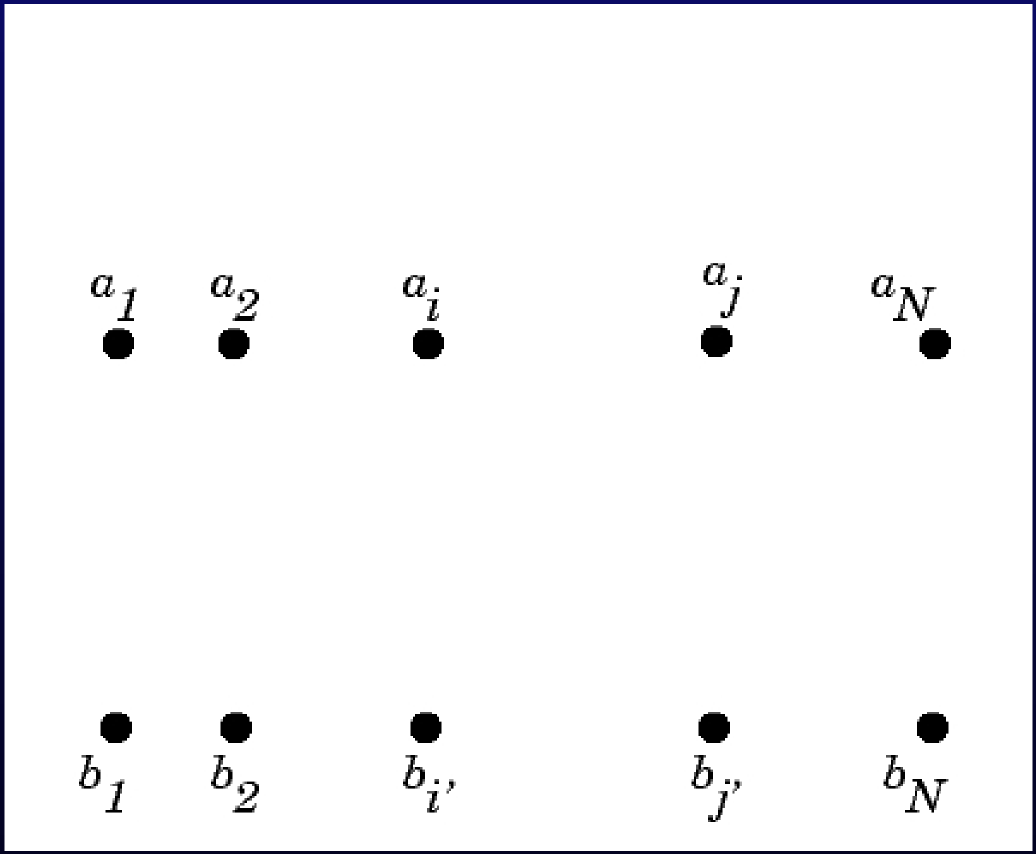
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.



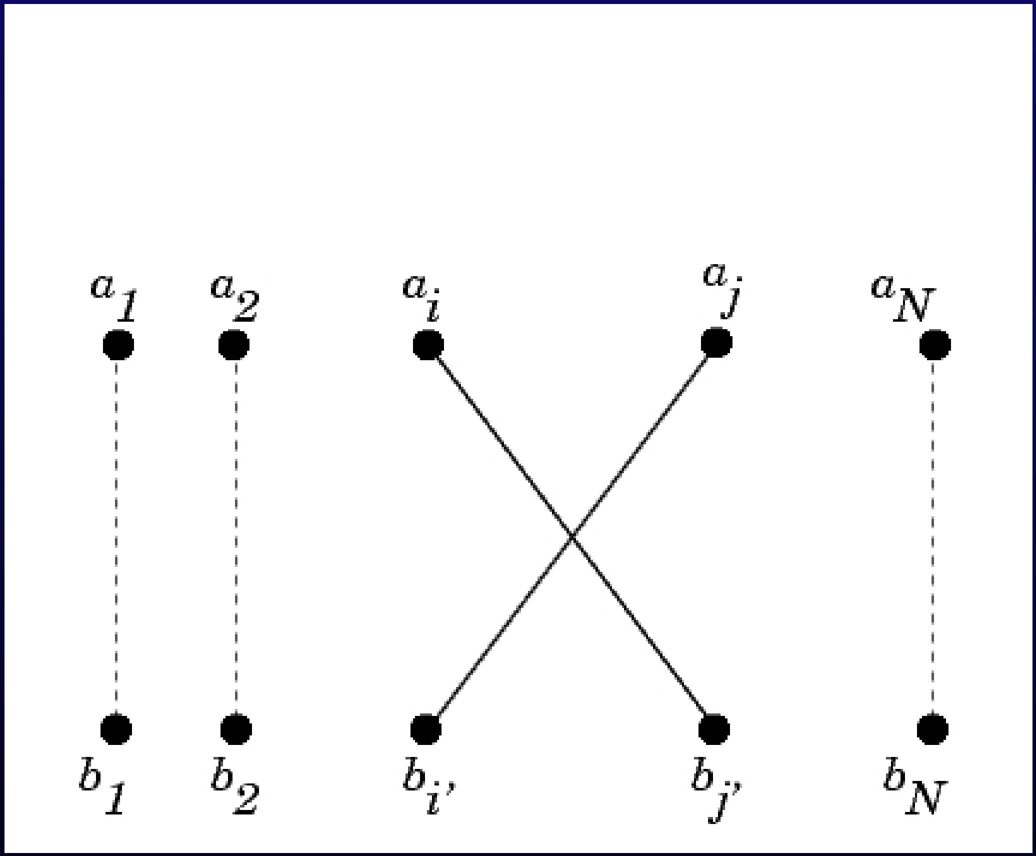
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

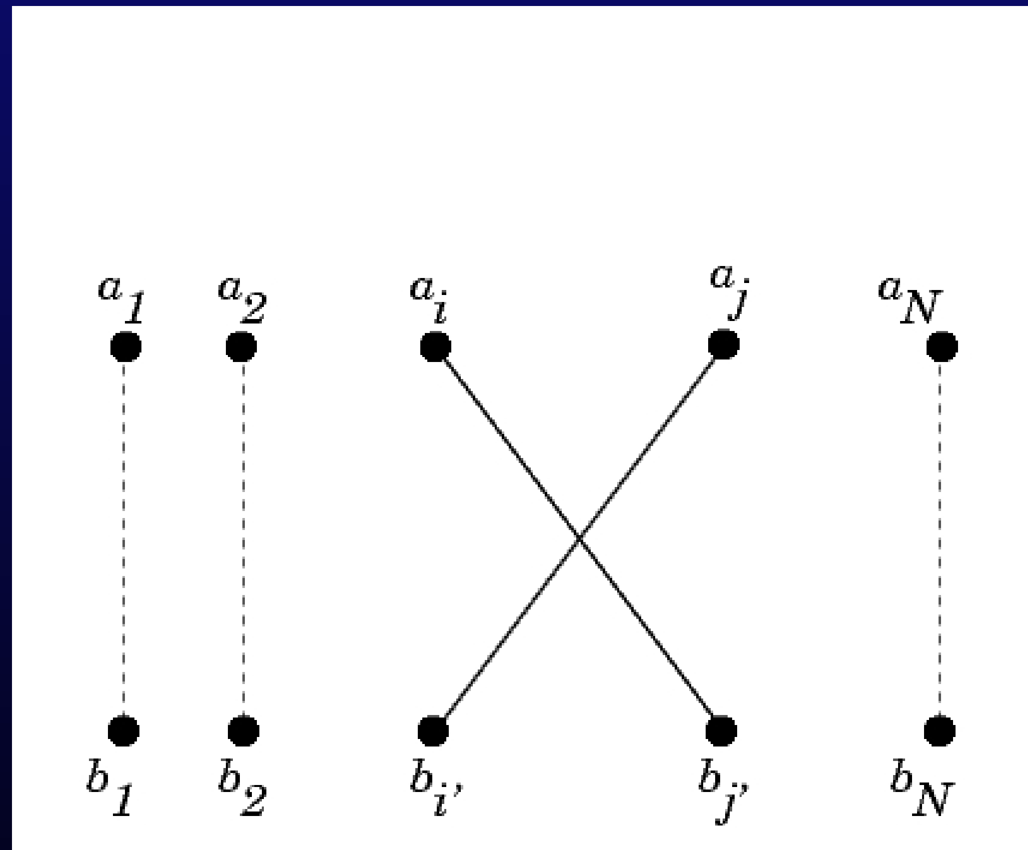
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

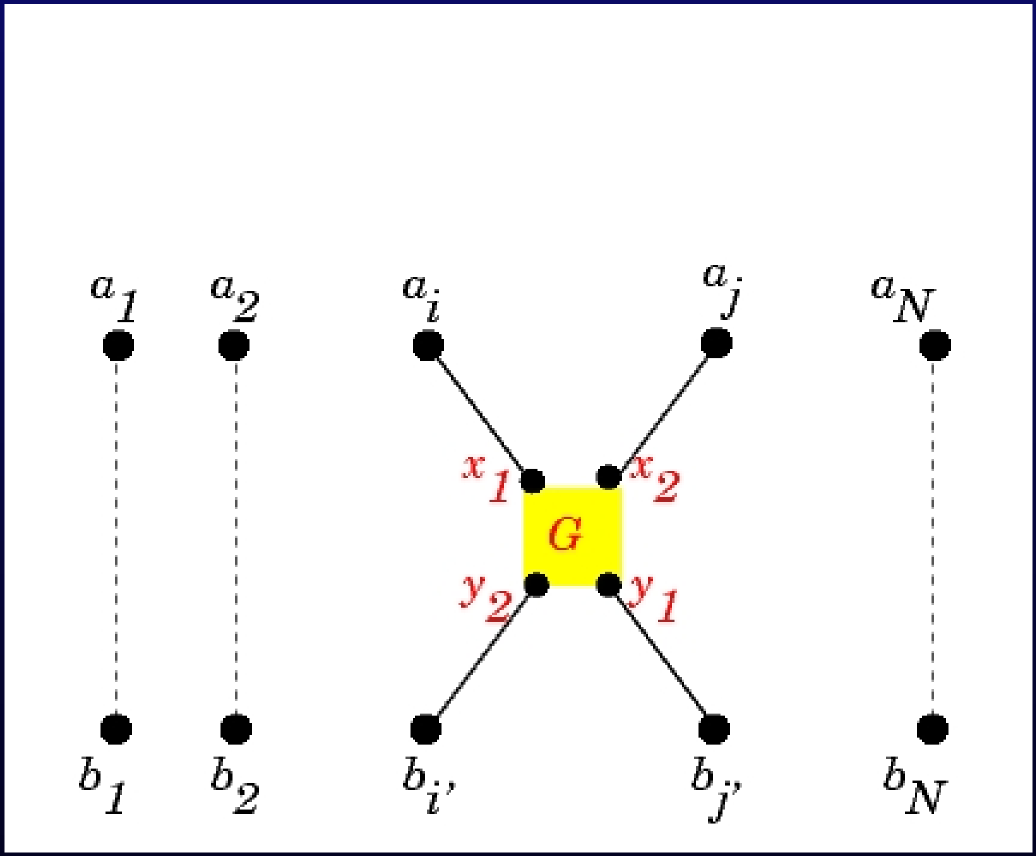
At each intersection put a copy of G.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

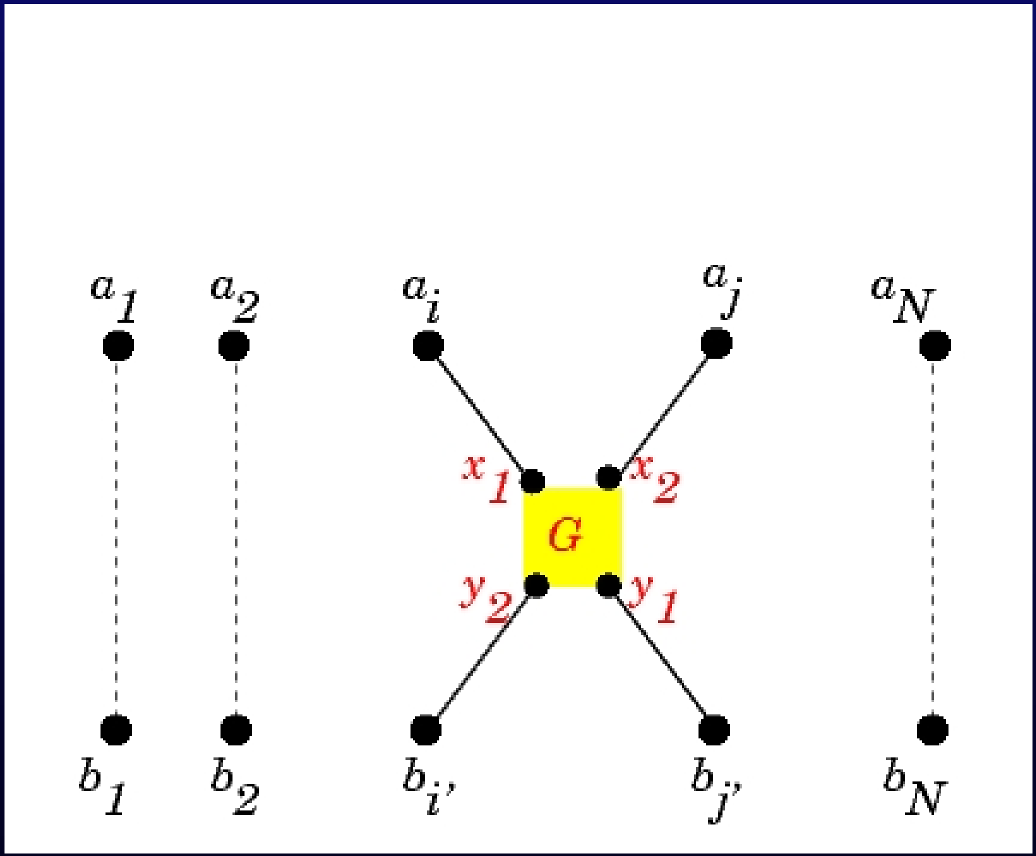


With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

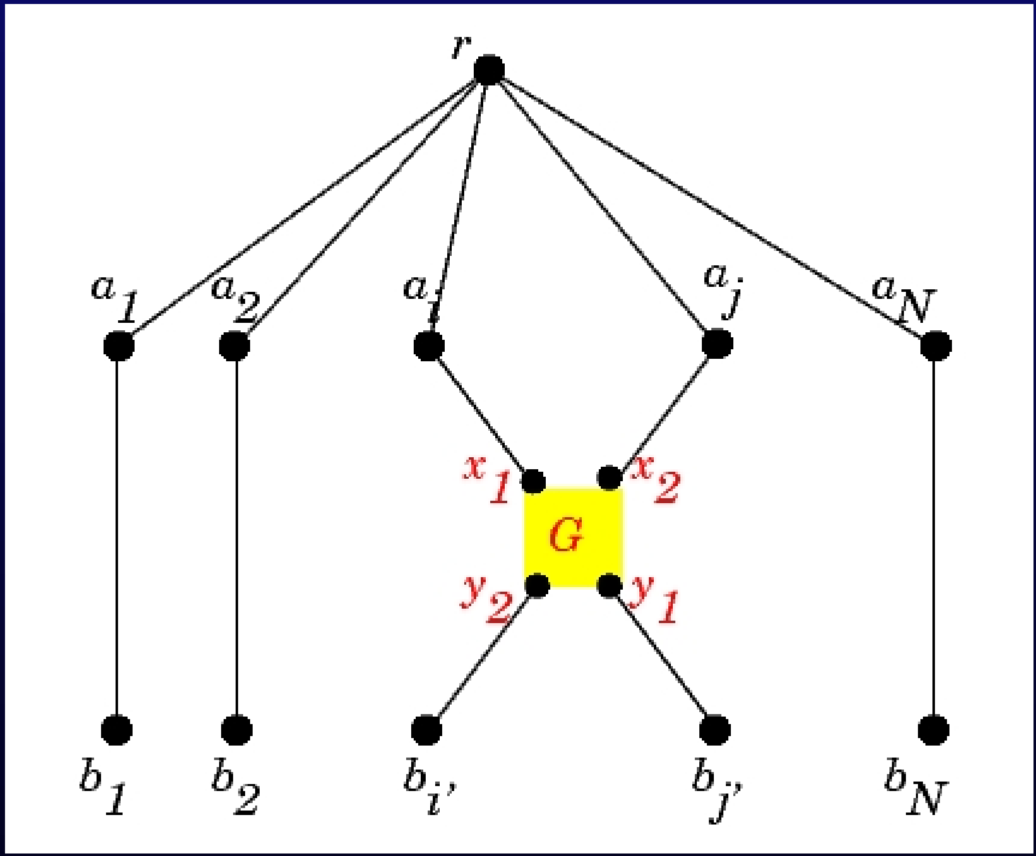


With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$



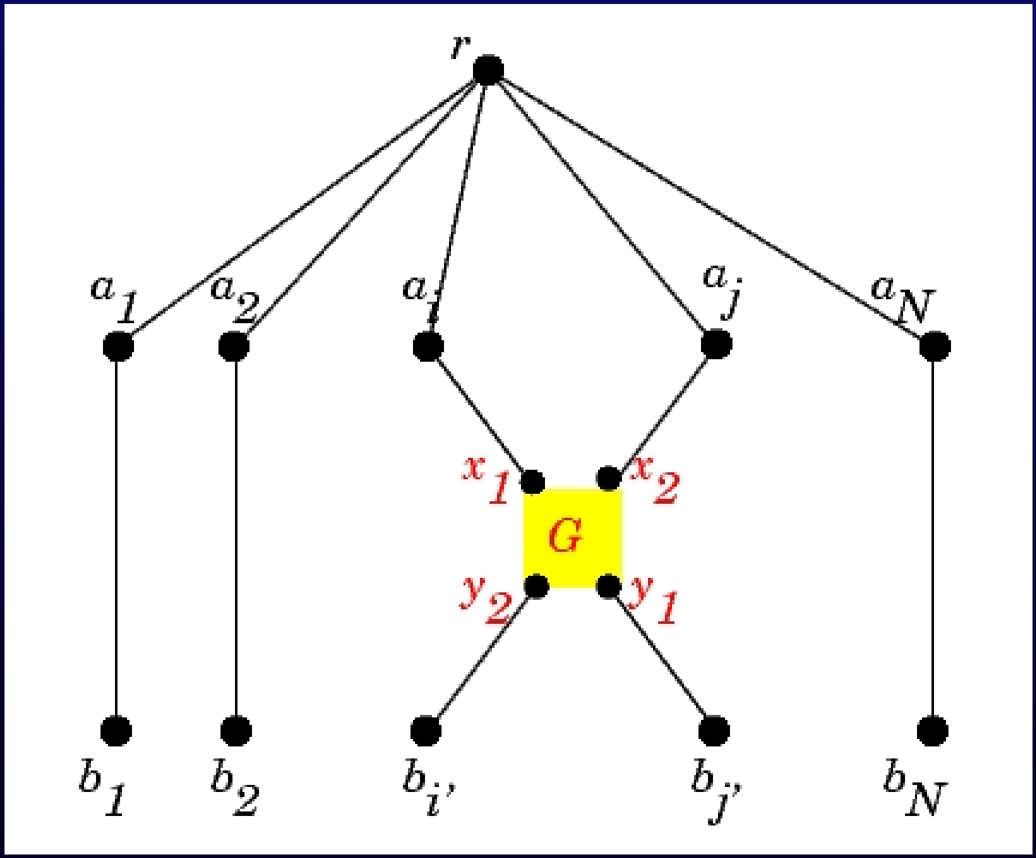
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

All edges are directed top to bottom. Let $T = r \cup \{b_1, \ldots, b_N\}$.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

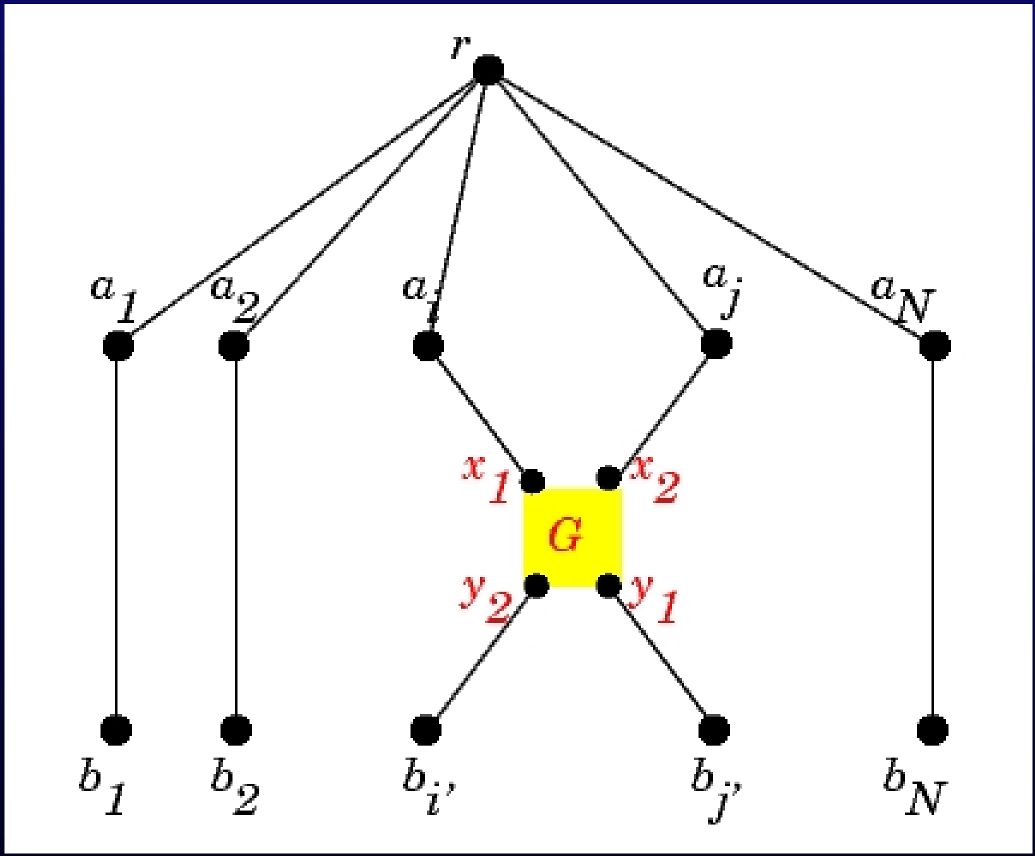
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

All edges are directed top to bottom. Let $T = r \cup \{b_1, \ldots, b_N\}$.

Lemma 1: If G is a *"yes"* instance of 2DIRPATH then H has N edge-disjoint Steiner trees.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

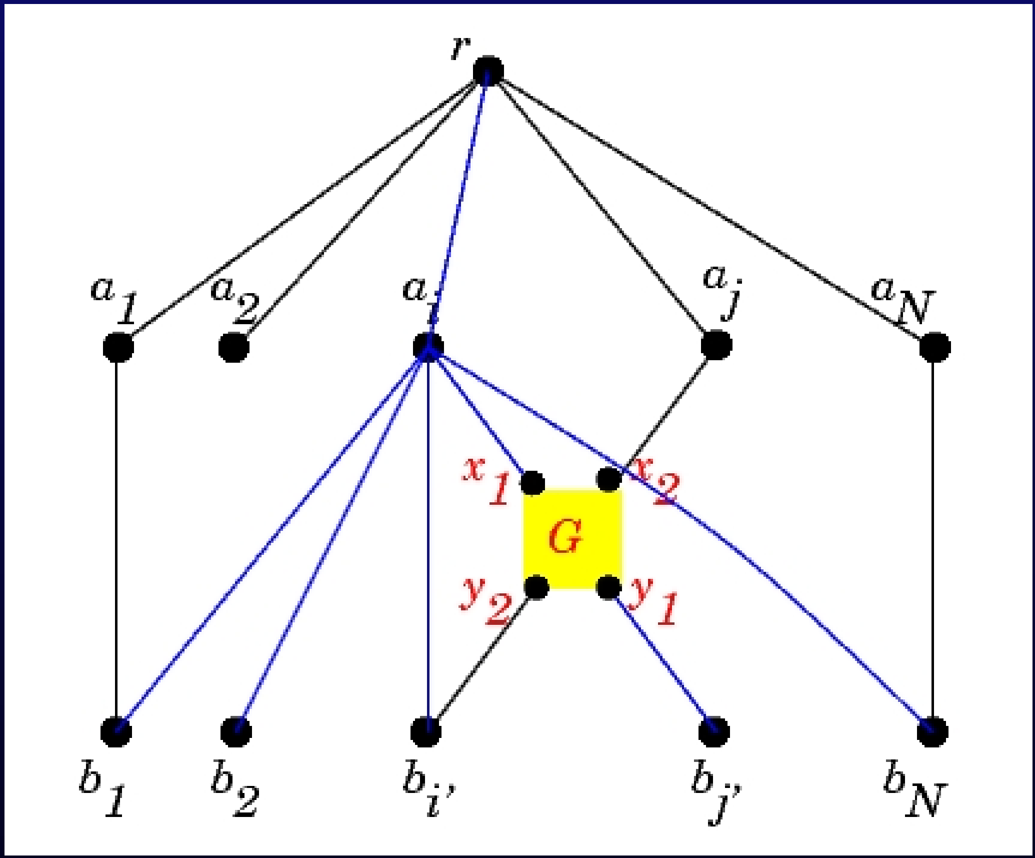
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

All edges are directed top to bottom. Let $T = r \cup \{b_1, \ldots, b_N\}$.

Lemma 1: If G is a *"yes"* instance of 2DIRPATH then H has N edge-disjoint Steiner trees.



With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

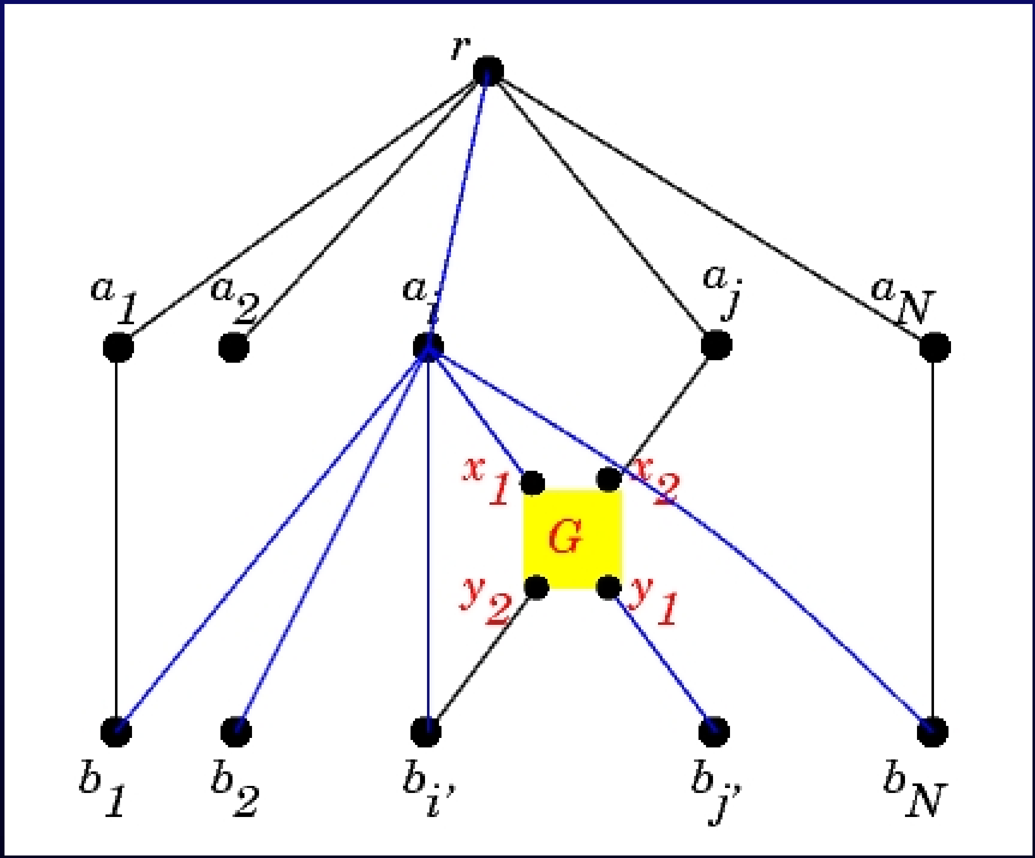
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

All edges are directed top to bottom. Let $T = r \cup \{b_1, \ldots, b_N\}$.

Lemma 1: If G is a *"yes"* instance of 2DIRPATH then H has N edge-disjoint Steiner trees.



Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

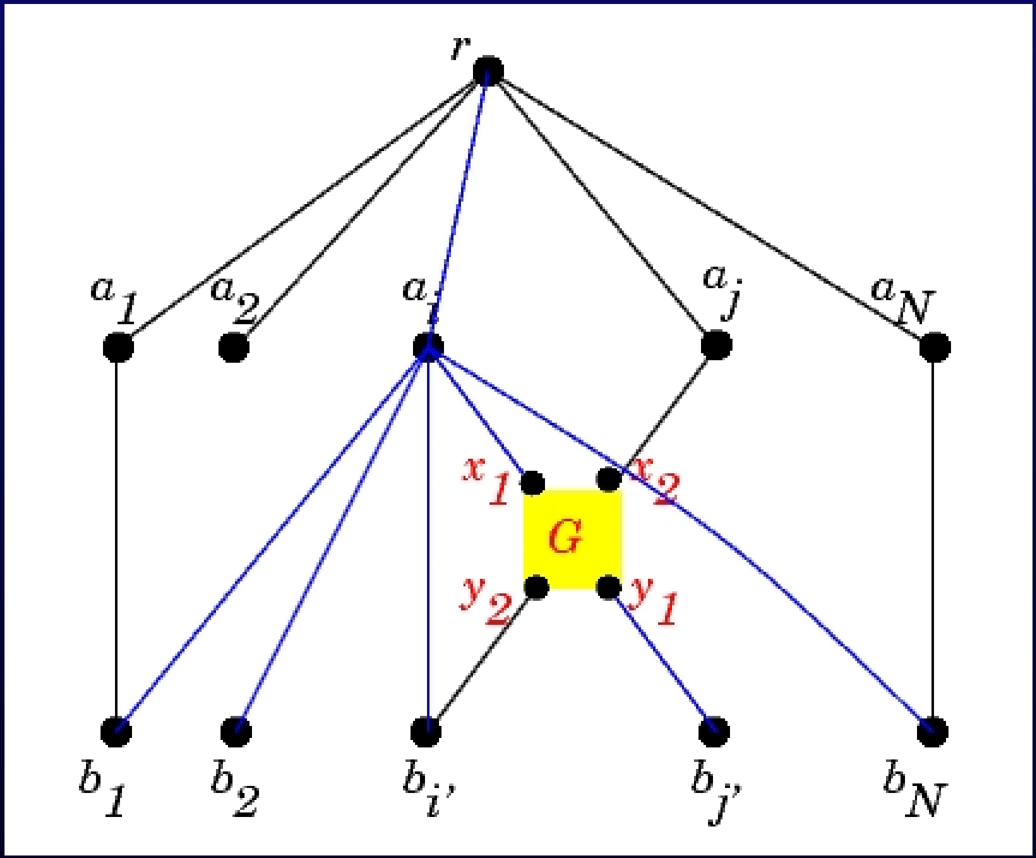
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

At each intersection put a copy of G.

Create a root r and connect it to $\{a_1, \ldots, a_N\}$, and now put edges $a_i b_i$, for $1 \leq i \leq N$

All edges are directed top to bottom. Let $T = r \cup \{b_1, \ldots, b_N\}$.



Lemma 1: If G is a *"yes"* instance of 2DIRPATH then H has N edge-disjoint Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Thus deciding between 1 and N Steiner trees in H is NP-hard.

Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

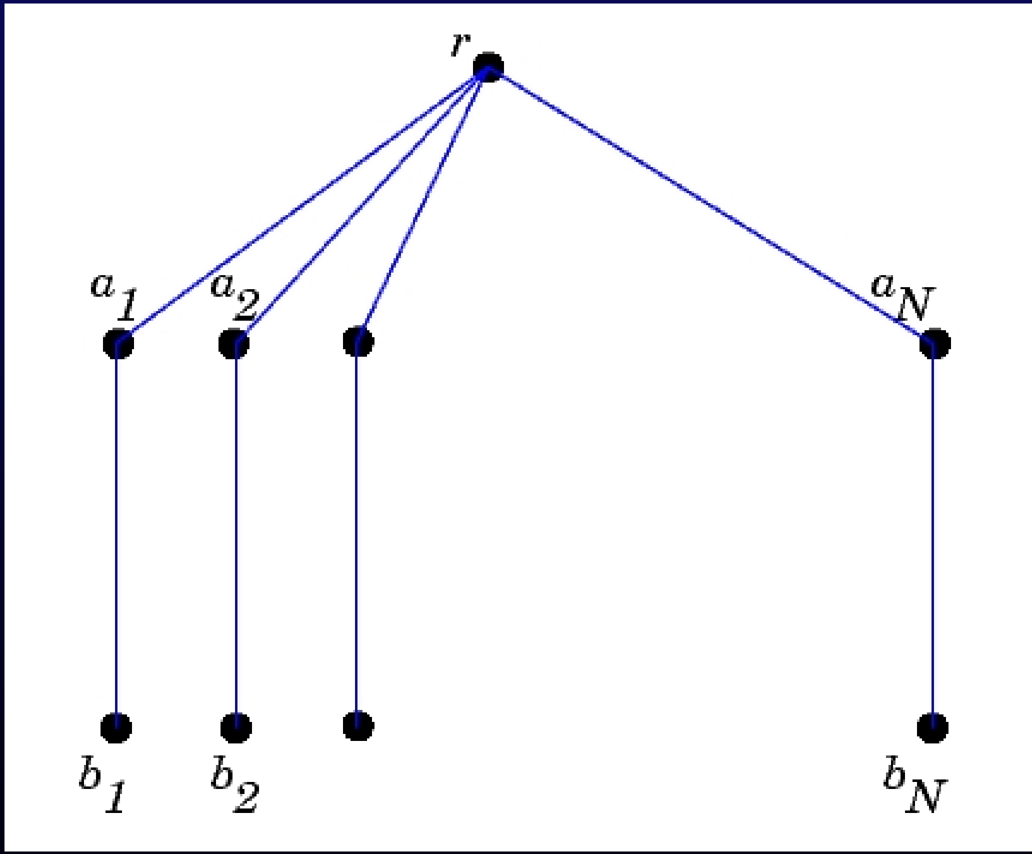
Proof: First note that H has at least one Steiner tree.

Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.

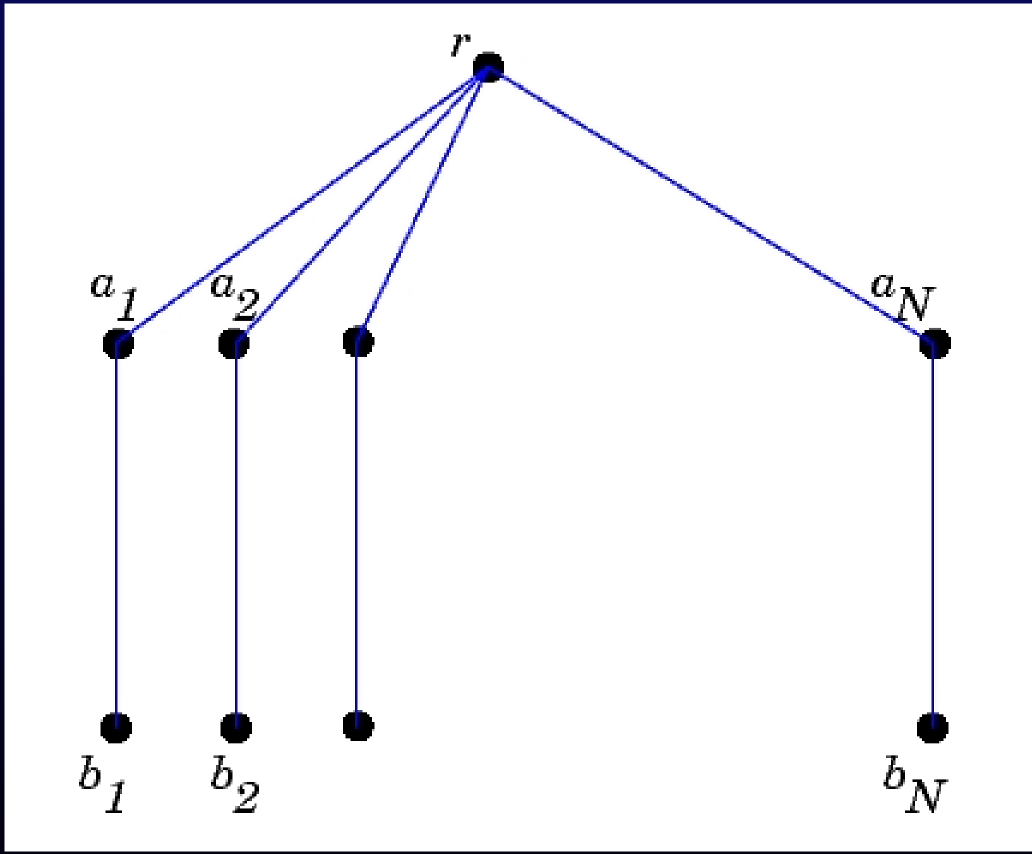


Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.



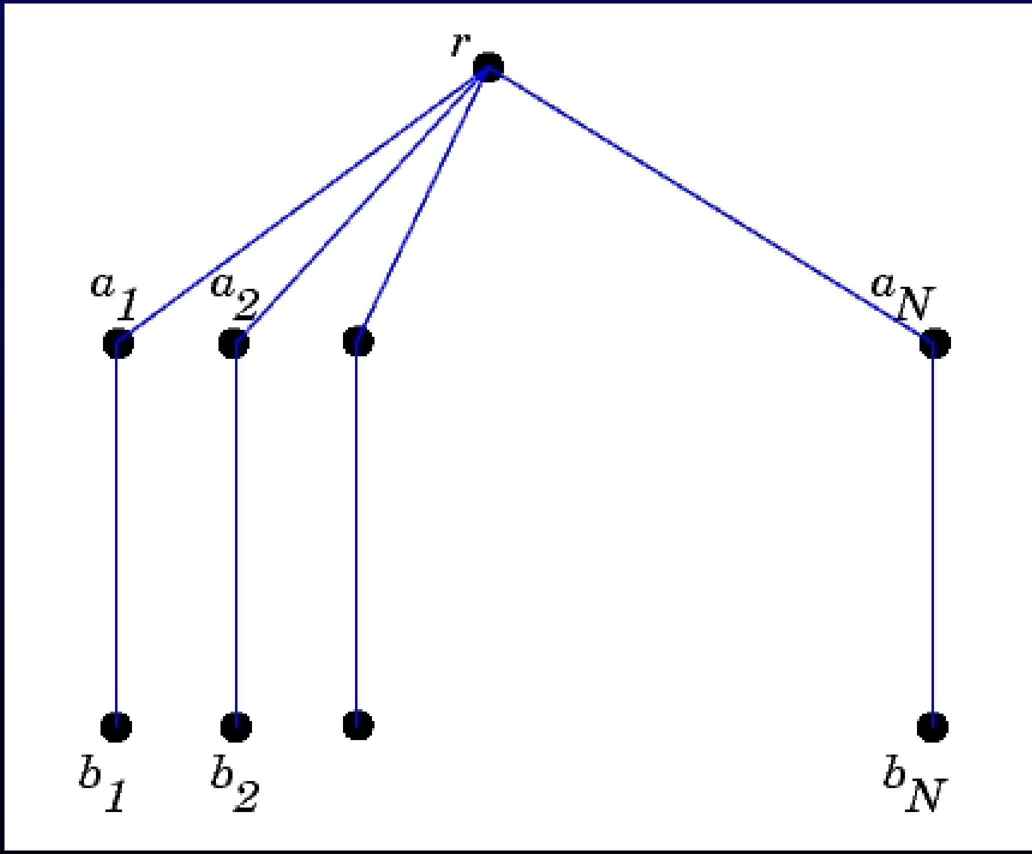
Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.

Important observation: There is no $a_1 b_{j\geq 2}$ path in any tree in $\mathcal{T}$, else if there is such path in, say T_1,



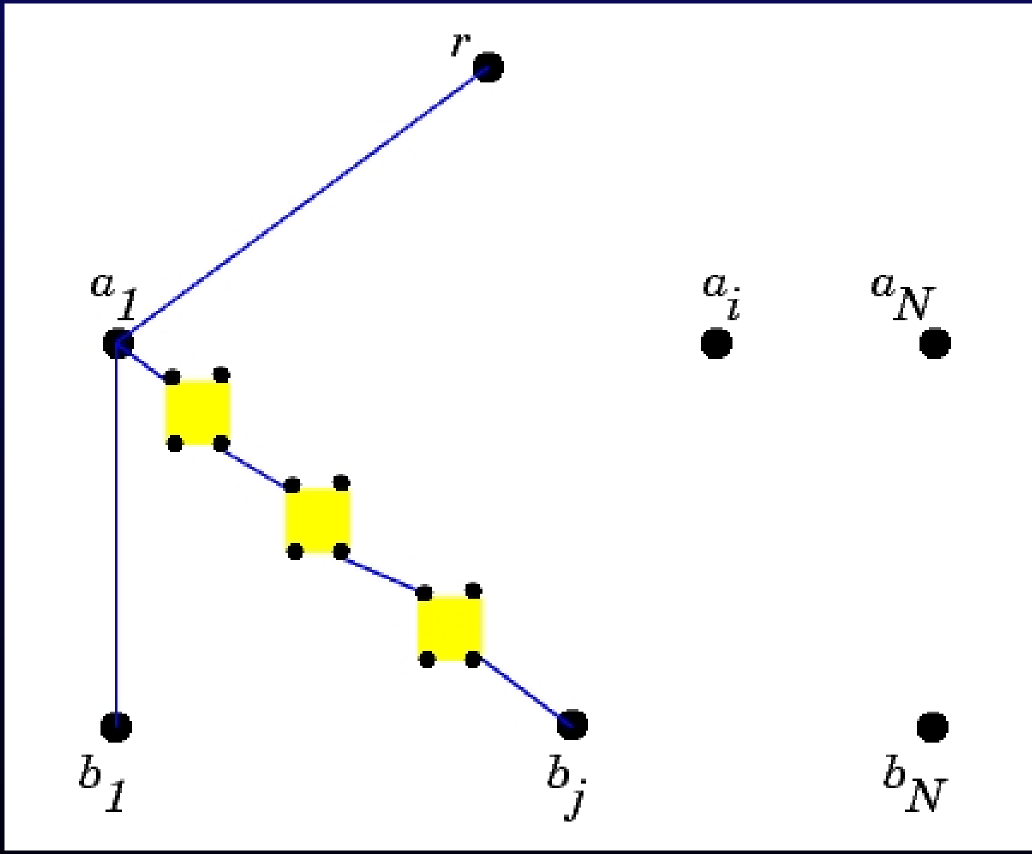
Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.

Important observation: There is no $a_1 b_{j\geq 2}$ path in any tree in $\mathcal{T}$, else if there is such path in, say T_1,

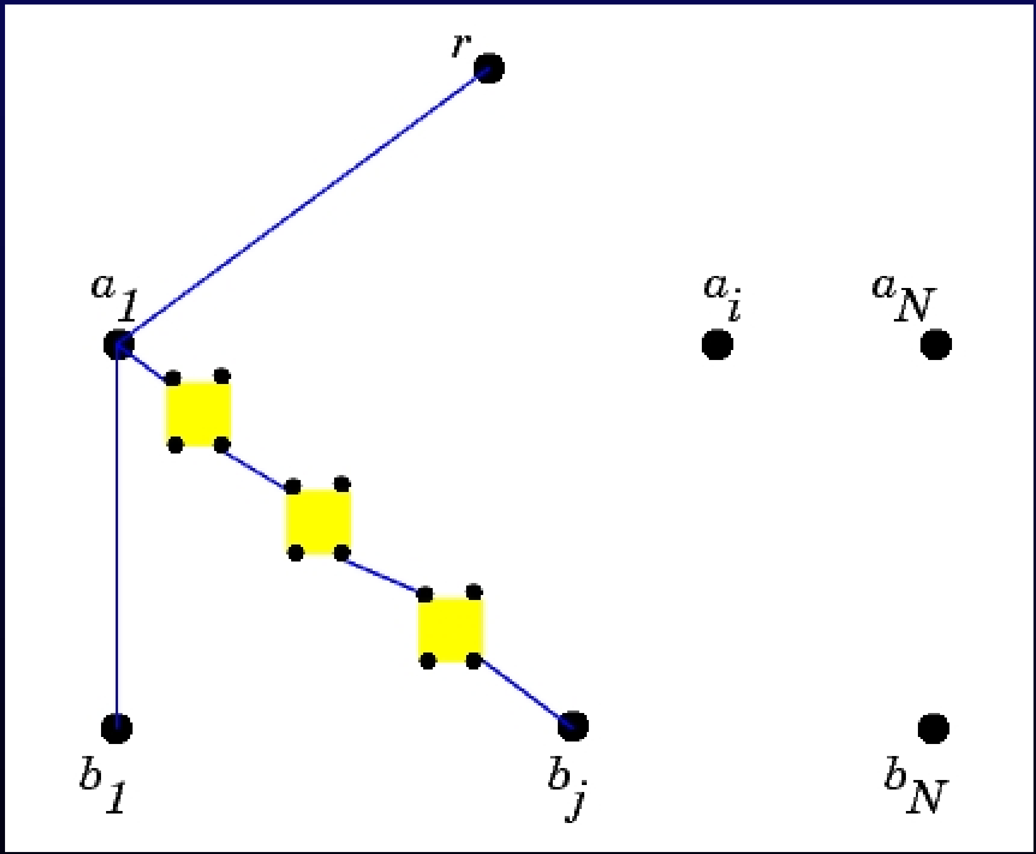


Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.



Important observation: There is no $a_1 b_{j \geq 2}$ path in any tree in $\mathcal{T}$, else if there is such path in, say T_1, then there cannot be a path to b_1 in any other tree.

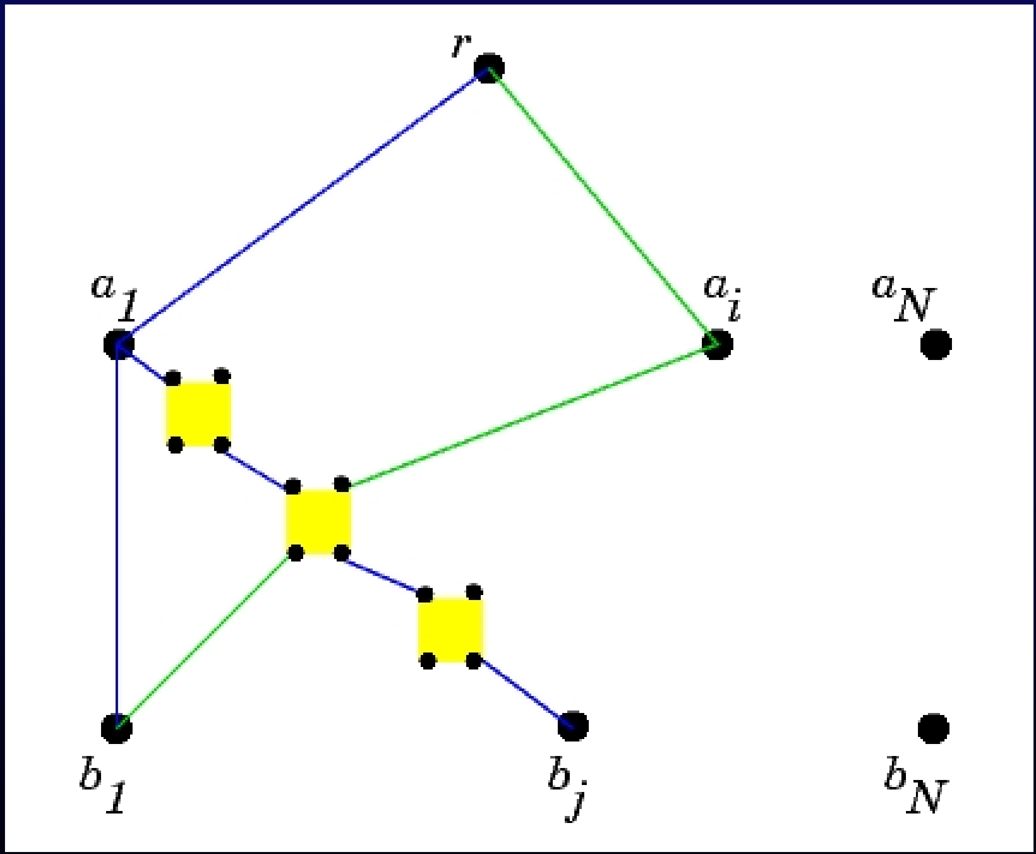
Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.

Important observation: There is no $a_1 b_{j\geq 2}$ path in any tree in $\mathcal{T}$, else if there is such path in, say T_1, then there cannot be a path to b_1 in any other tree.



By induction, we can show there is no $a_i b_j$ path, $i < j$, in any tree of $\mathcal{T}$.

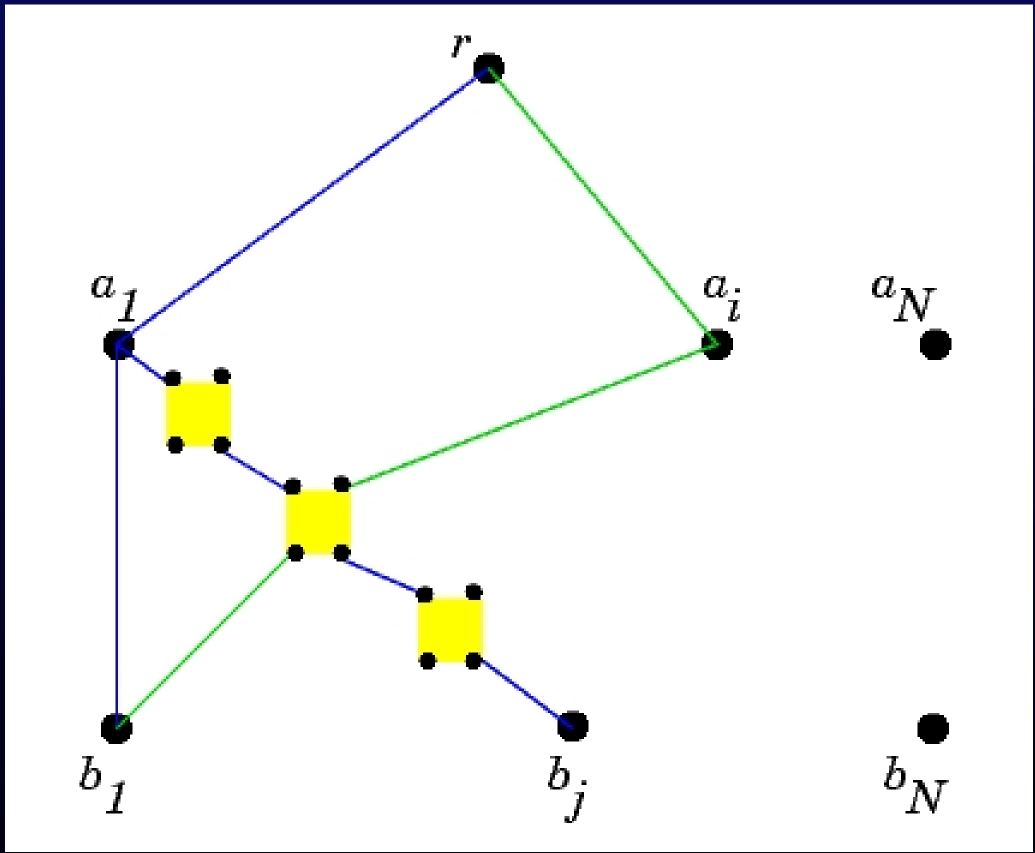
Since H has $O(N^4)$ copies of G and $N = |E(G)|^{\frac{1}{\epsilon}}$: $m = E(H) = O(N^{4+\epsilon})$.

So it is NP-hard to decide between 1 and $O(m^{\frac{1}{4}-\epsilon'})$ Steiner trees.

Lemma 2: If G is a *"no"* instance of 2DIRPATH then H has no more than 1 edge-disjoint Steiner tree.

Proof: First note that H has at least one Steiner tree.
Suppose that G is a *"no"* instance and $\mathcal{T} = \{T_1, \ldots, T_k\}$ are edge-disjoint Steiner trees in H, with $k > 1$.

Important observation: There is no $a_1 b_{j\geq 2}$ path in any tree in $\mathcal{T}$, else if there is such path in, say T_1, then there cannot be a path to b_1 in any other tree.



By induction, we can show there is no $a_i b_j$ path, $i < j$, in any tree of $\mathcal{T}$.

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Back to the undirected setting

we showed that both PEU and PVU are APX-hard even for constant number of terminals and we have constant approximation for PEU.

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Back to the undirected setting

we showed that both PEU and PVU are APX-hard even for constant number of terminals and we have constant approximation for PEU.

What about approximation algorithms for PVU?

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Back to the undirected setting

we showed that both PEU and PVU are APX-hard even for constant number of terminals and we have constant approximation for PEU.

What about approximation algorithms for PVU?

We prove that PVU is significantly harder than PEU:

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Back to the undirected setting

we showed that both PEU and PVU are APX-hard even for constant number of terminals and we have constant approximation for PEU.

What about approximation algorithms for PVU?

We prove that PVU is significantly harder than PEU:

Theorem (Cheriyan & S.): PVU cannot be approximated with ratio $(1-\epsilon)\ln n$, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\log\log n})$.

Using a similar reduction, this time using 2DIRPATH for vertex-disjoint paths:

Theorem (Cheriyan & S.): Unless $P = NP$, every approx algorithm for PVD has factor $\Omega(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

On the other hand, an algorithm similar to the one presented for PED yields:

Theorem (Cheriyan & S.): There is a polynomial time $O(n^{\frac{1}{2}+\epsilon})$-approximation algorithm for PVD.

Back to the undirected setting

we showed that both PEU and PVU are APX-hard even for constant number of terminals and we have constant approximation for PEU.

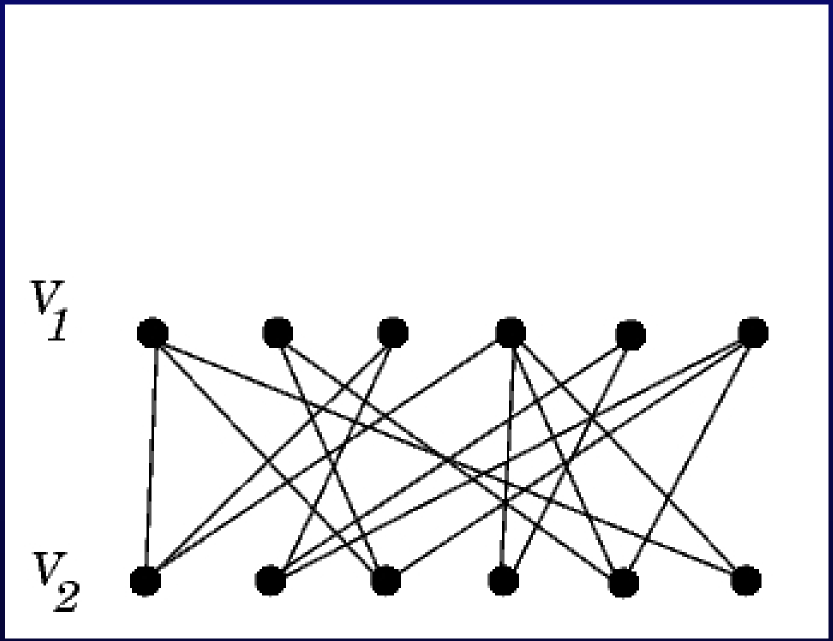
What about approximation algorithms for PVU?

We prove that PVU is significantly harder than PEU:

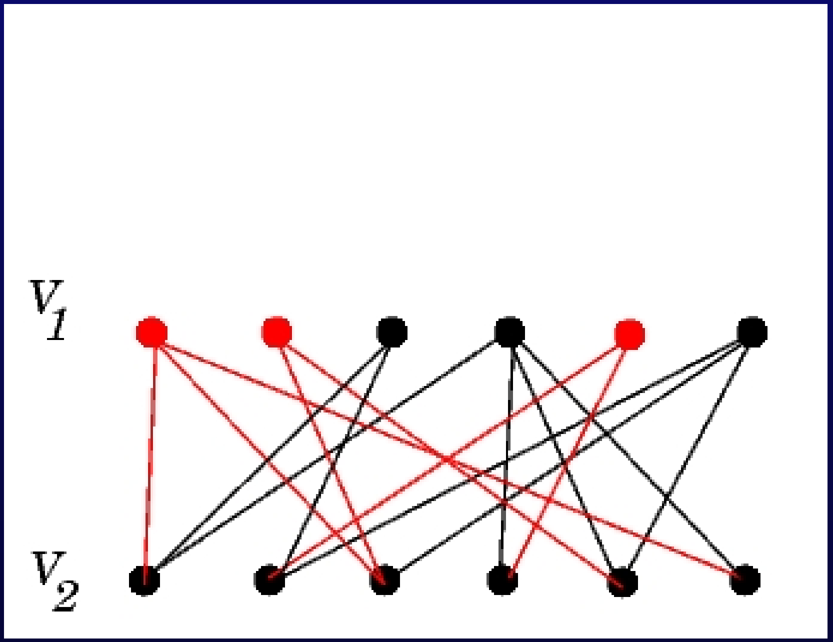
Theorem (Cheriyan & S.): PVU cannot be approximated with ratio $(1-\epsilon)\ln n$, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\log\log n})$.

proof: Reduction from Set-Cover Packing.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

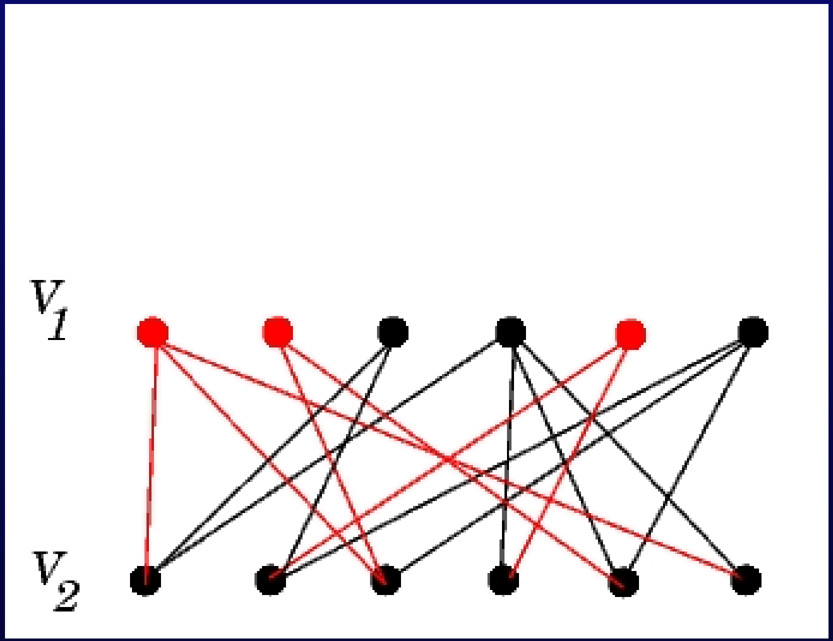


Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.



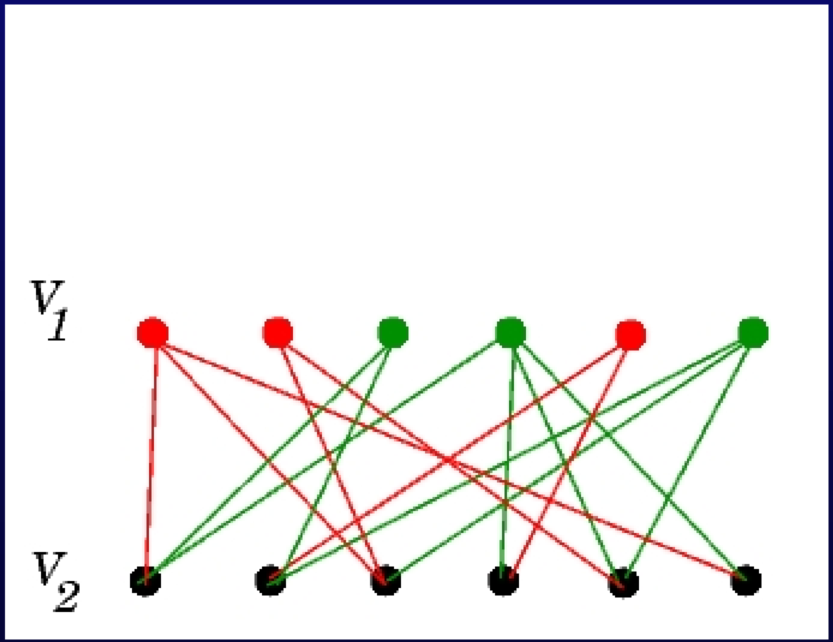
Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.



Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

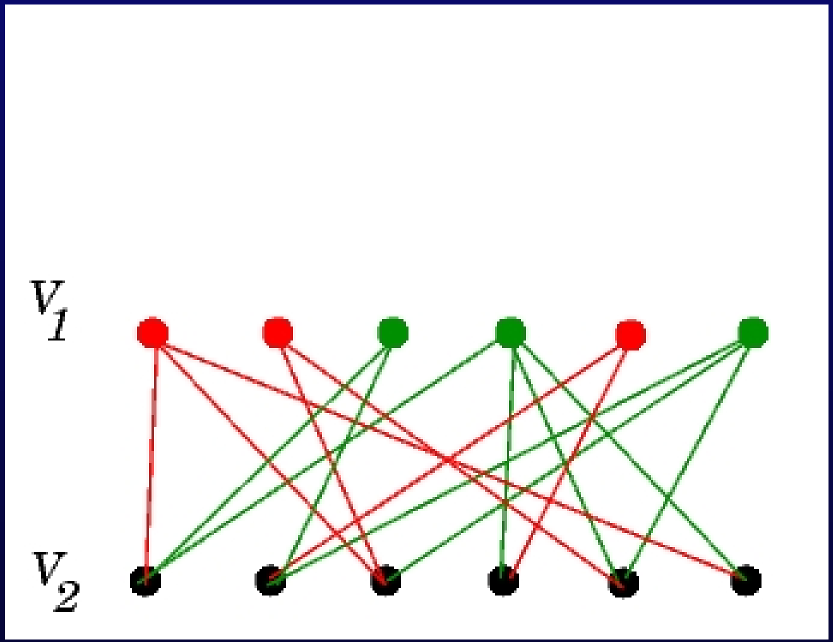
Goal: Find max number of disjoint set-covers of V_2.



Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

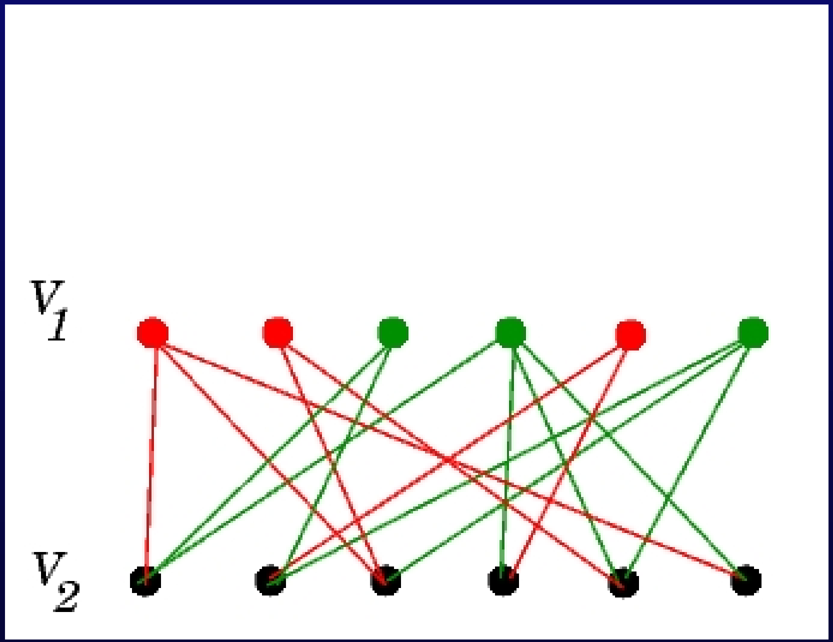
Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.



Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

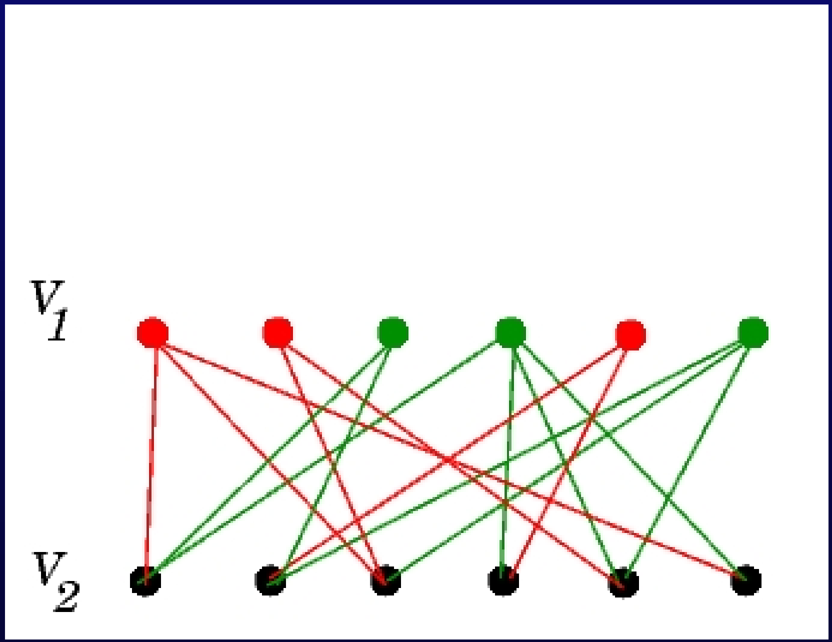
Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.



Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.



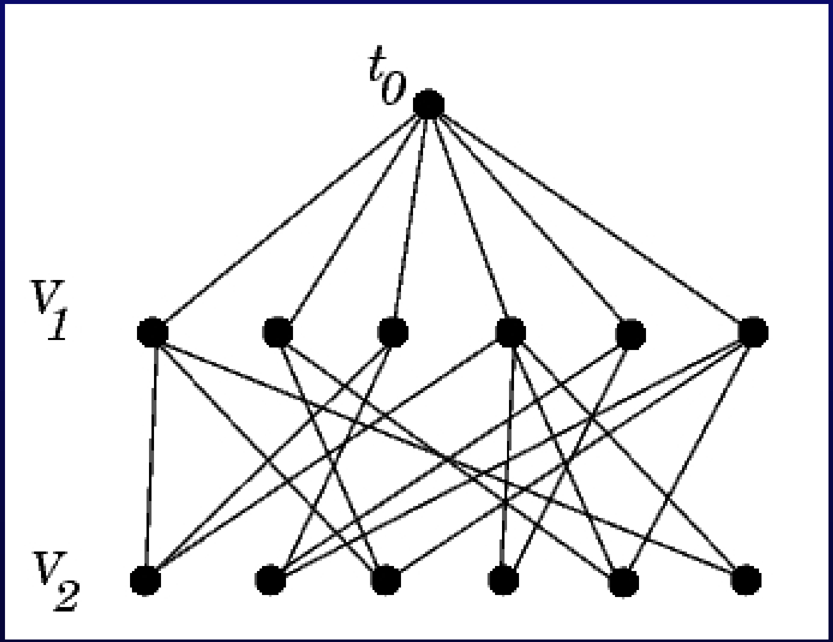
Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.

Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log \log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.

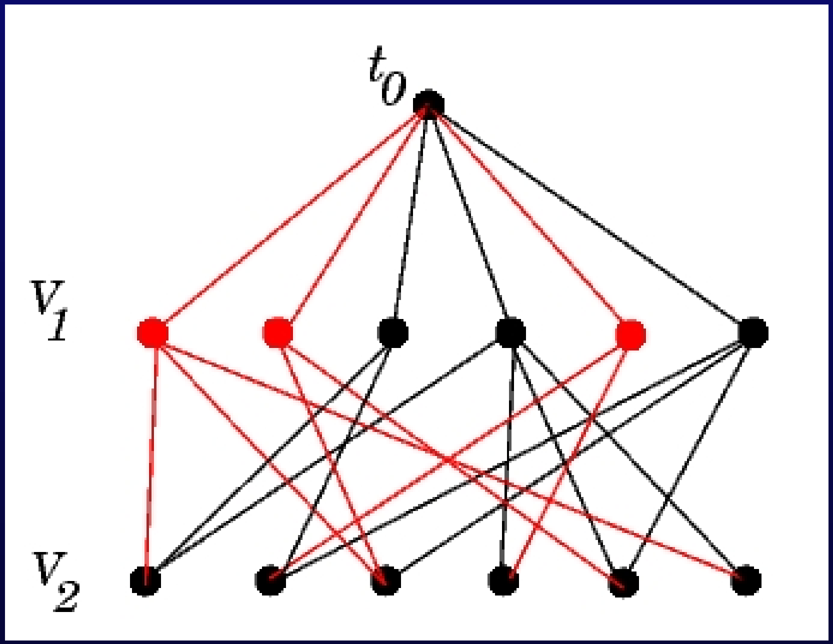


Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.

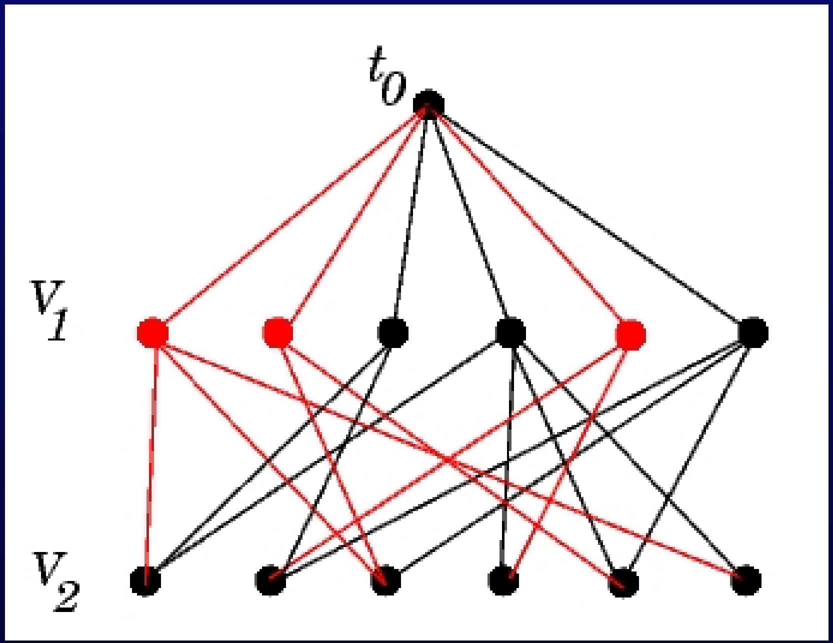


Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.



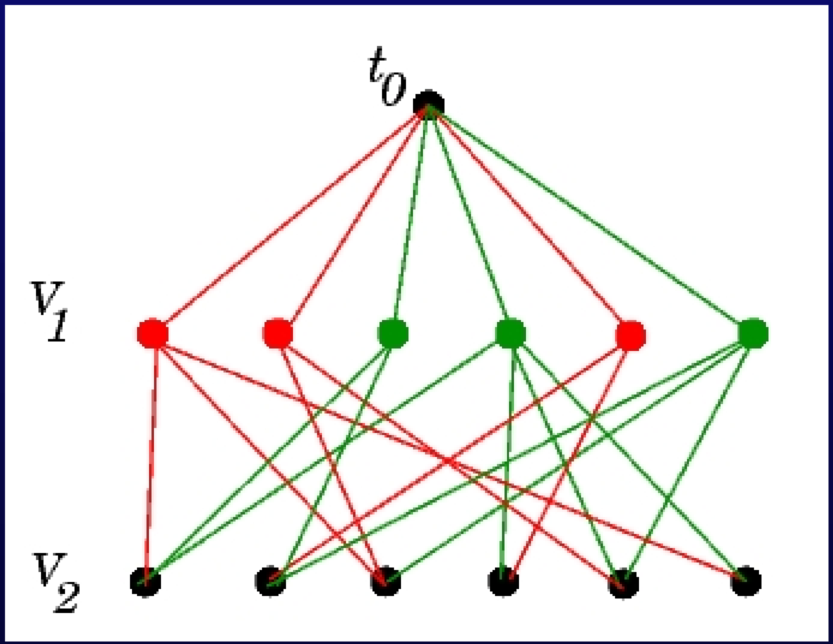
Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

If $S_1, \ldots, S_p$ form a set-cover then $T_i = t_0 \cup S_i \cup V_2$ (for $1 \le i \le p$ form a set of V.D. Steiner trees.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.



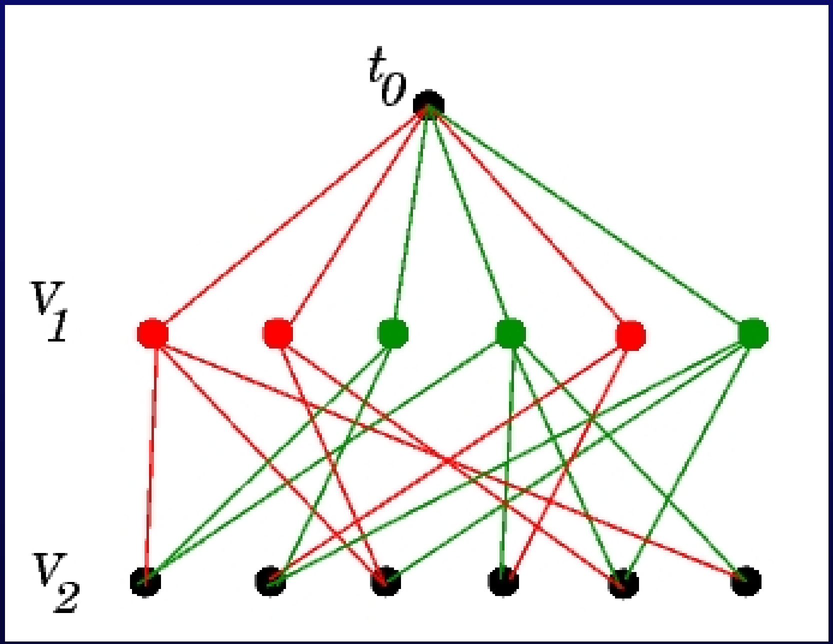
Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

If $S_1, \ldots, S_p$ form a set-cover then $T_i = t_0 \cup S_i \cup V_2$ (for $1 \le i \le p$ form a set of V.D. Steiner trees.

Set-Cover packing: Given bipartite graph $G(V_1 \cup V_2, E)$, a set-cover of V_2 is a subset $S \subseteq V_1$ s.t. covers V_2.

Goal: Find max number of disjoint set-covers of V_2.

Theorem (Fiege, Halldorson, Kortsarz, Srinivasan'02): Unless $NP \subseteq DTIME(n^{\log\log n})$ there is no $(1-\epsilon)\ln n$ approximation algorithm (for any $\epsilon > 0$) for set-cover packing.



Given $G(V_1 \cup V_2, E)$, add vertex t_0 and connect to V_1. Let $T = V_2 \cup \{t_0\}$.

If $S_1, \ldots, S_p$ form a set-cover then $T_i = t_0 \cup S_i \cup V_2$ (for $1 \leq i \leq p$ form a set of V.D. Steiner trees.

Conversely, if $T_1, \ldots, T_p$ are V.D. Steiner trees, because V_2 is independent set, there is a set $S_i \subset V(T_i)$ s.t. covers V_2.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$

Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n \sqrt{n})$.

The algorithm is similar to those for PED and PVD:

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.
- The separation oracle for dual is minimum node-weighted Steiner tree problem.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.
- The separation oracle for dual is minimum node-weighted Steiner tree problem.

  Theorem (Guha & Khuller'03): Min. node-weighted Steiner tree can be approximated within $O(\log n)$.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.
- The separation oracle for dual is minimum node-weighted Steiner tree problem.

  Theorem (Guha & Khuller'03): Min. node-weighted Steiner tree can be approximated within $O(\log n)$.
- We can also prove:
  Theorem: There is an α-approx algorithm for fractional PVU iff there is an α-approx algorithm for min node-weighted Steiner tree.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.
- The separation oracle for dual is minimum node-weighted Steiner tree problem.

  Theorem (Guha & Khuller'03): Min. node-weighted Steiner tree can be approximated within $O(\log n)$.
- We can also prove:
  Theorem: There is an α-approx algorithm for fractional PVU iff there is an α-approx algorithm for min node-weighted Steiner tree.$\Longrightarrow$

  Corollary: There is an $O(\log n)$ approx algorithm for fractional PVU.

Approximation algorithm for PVU

Bad news: PVU is hard to approximate within $O(\log n)$
Good news: We can approximate PVU within $O(\log n\sqrt{n})$.

The algorithm is similar to those for PED and PVD:

- Formulate PVU as an ILP, relax it to an LP, and consider the dual.
- The separation oracle for dual is minimum node-weighted Steiner tree problem.

  Theorem (Guha & Khuller'03): Min. node-weighted Steiner tree can be approximated within $O(\log n)$.
- We can also prove:
  Theorem: There is an α-approx algorithm for fractional PVU iff there is an α-approx algorithm for min node-weighted Steiner tree.$\Longrightarrow$

  Corollary: There is an $O(\log n)$ approx algorithm for fractional PVU.
- Use randomized rounding to get an $O(\log n\sqrt{n})$ approximation.

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$
PVU	$O(\log n\sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1 + \epsilon_0$
PVU	$O(\log n\sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1 + \epsilon_0$
PVU	$O(\log n \sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.
- We know PEU with 4 terminals is APX-hard. What about 3 terminals? Is it NP-complete?

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$
PVU	$O(\log n\sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.
- We know PEU with 4 terminals is APX-hard. What about 3 terminals? Is it NP-complete?
- The gap for PVU is not even within the same class!

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1 + \epsilon_0$
PVU	$O(\log n \sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.
- We know PEU with 4 terminals is APX-hard. What about 3 terminals? Is it NP-complete?
- The gap for PVU is not even within the same class!
  Is there an $O(\log^k n)$ approx. for PVU?

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1 + \epsilon_0$
PVU	$O(\log n \sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.
- We know PEU with 4 terminals is APX-hard. What about 3 terminals? Is it NP-complete?
- The gap for PVU is not even within the same class!
  Is there an $O(\log^k n)$ approx. for PVU?
- What is the integrality gap for PVU?

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$
PVU	$O(\log n\sqrt{n})$	$\Omega(\log n)$
PED	$O(m^{\frac{1}{2}+\epsilon})$	$\Omega(m^{\frac{1}{3}-\epsilon})$
PVD	$O(n^{\frac{1}{2}+\epsilon})$	$\Omega(n^{\frac{1}{3}-\epsilon})$

- Close the gap for PEU.
- We know PEU with 4 terminals is APX-hard. What about 3 terminals? Is it NP-complete?
- The gap for PVU is not even within the same class!
  Is there an $O(\log^k n)$ approx. for PVU?
- What is the integrality gap for PVU?

Thanks!