Hardness and Approximation Results for Packing Steiner Trees

Mohammad R. Salavatipour

Department of Computing Science
University of Alberta

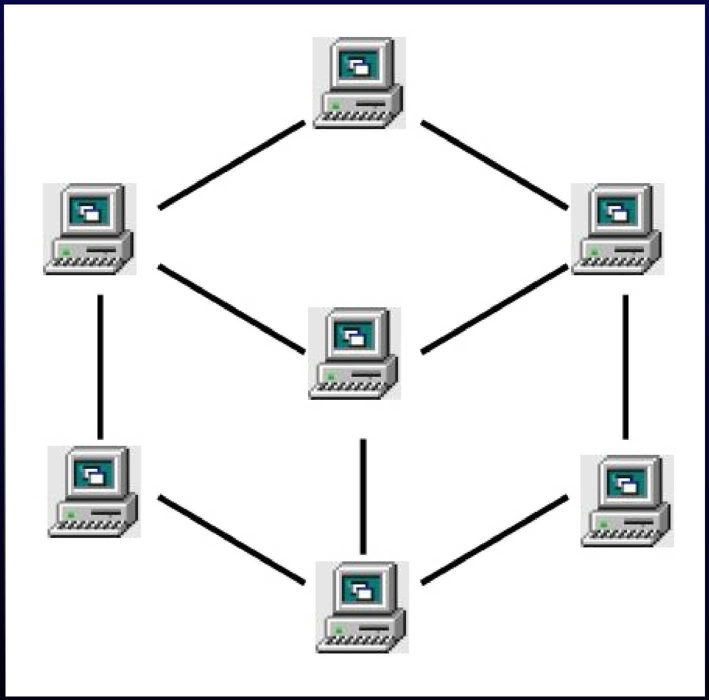
joint with

Joseph Cheriyan

Department of Combinatorics and Optimization
University of Waterloo

A Network Problem

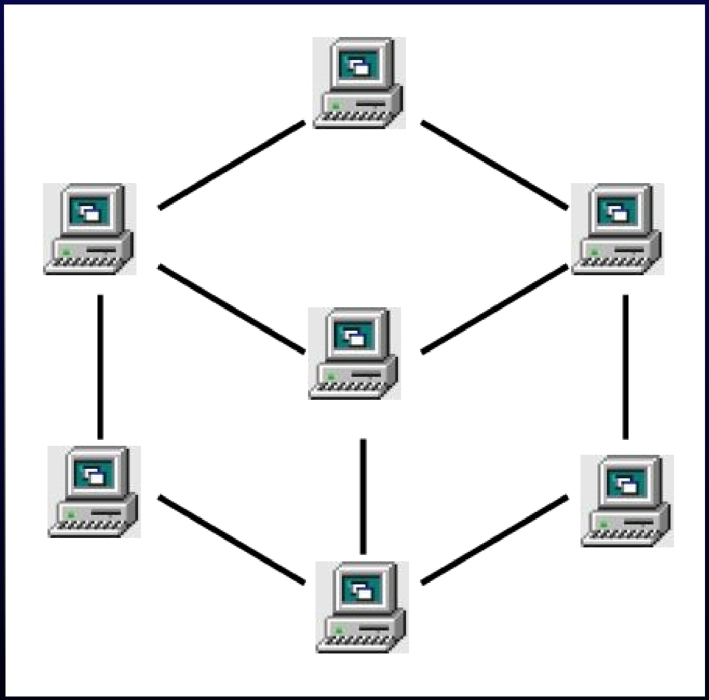
A network N with a *broadcaster*, we want to send some streams of video to some users



A Network Problem

A network N with a *broadcaster*, we want to send some streams of video to some users

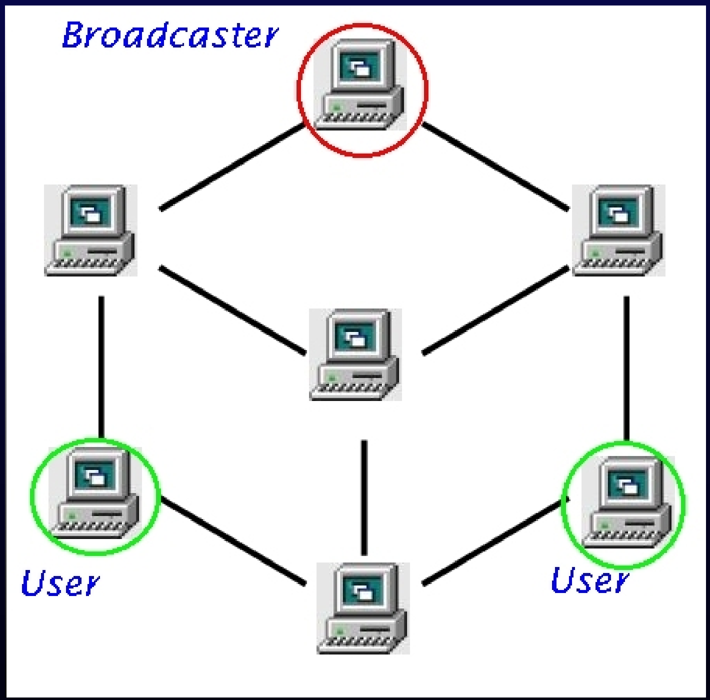
- *Users (terminals)*: a subset of nodes that have requested these streams,
- *Routers*: every node (including the users) can pass the data,



A Network Problem

A network N with a *broadcaster*, we want to send some streams of video to some users

- *Users (terminals)*: a subset of nodes that have requested these streams,
- *Routers*: every node (including the users) can pass the data,

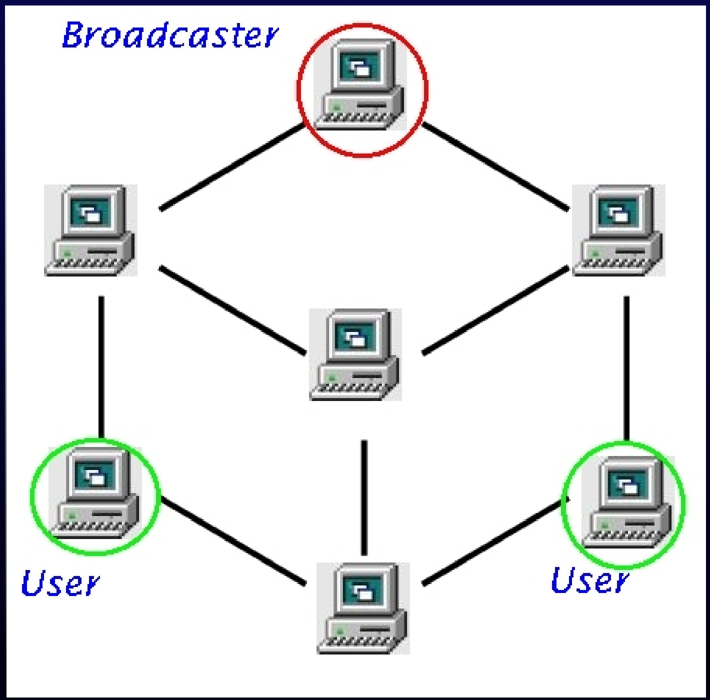


A Network Problem

A network N with a *broadcaster*, we want to send some streams of video to some users

- *Users (terminals)*: a subset of nodes that have requested these streams,
- *Routers*: every node (including the users) can pass the data,

Each stream of video traverses a tree in N that contains broadcaster and users, called Steiner tree.

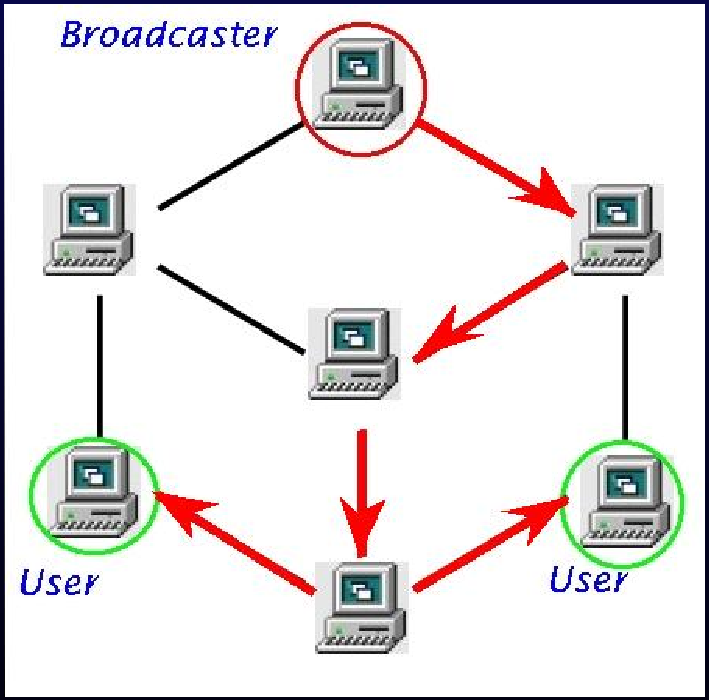


A Network Problem

A network N with a *broadcaster*, we want to send some streams of video to some users

- *Users (terminals)*: a subset of nodes that have requested these streams,
- *Routers*: every node (including the users) can pass the data,

Each stream of video traverses a tree in N that contains broadcaster and users, called Steiner tree.



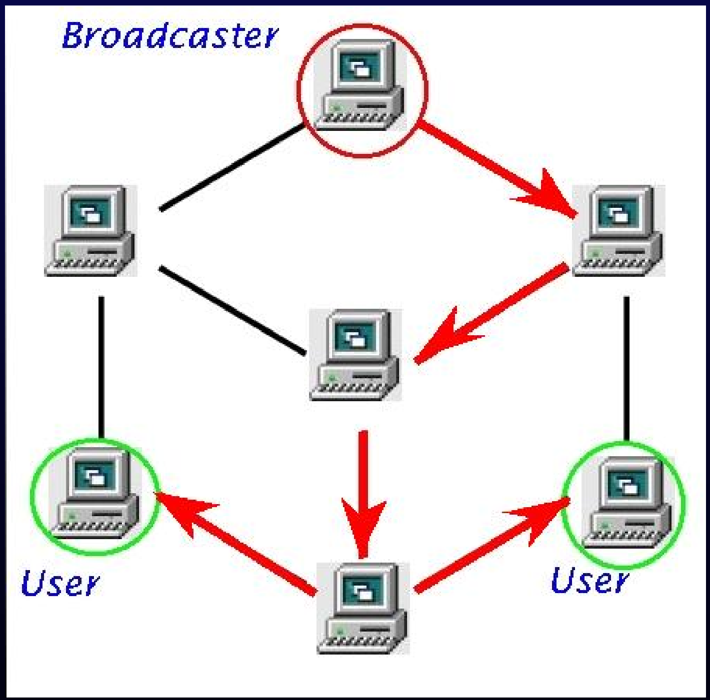
A Network Problem

A network N with a *broadcaster*, we want to send some streams of video to some users

- *Users (terminals)*: a subset of nodes that have requested these streams,
- *Routers*: every node (including the users) can pass the data,

Each stream of video traverses a tree in N that contains broadcaster and users, called Steiner tree.

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



To Graph Theory

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

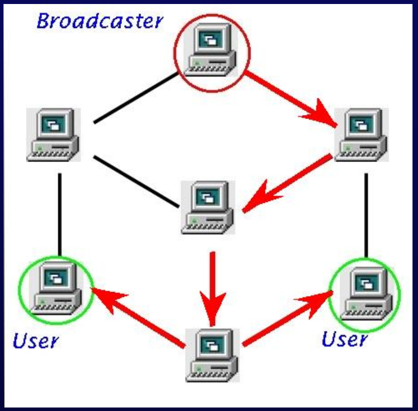
To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.

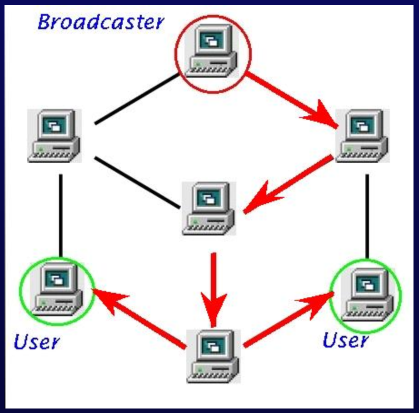
Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).



To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.

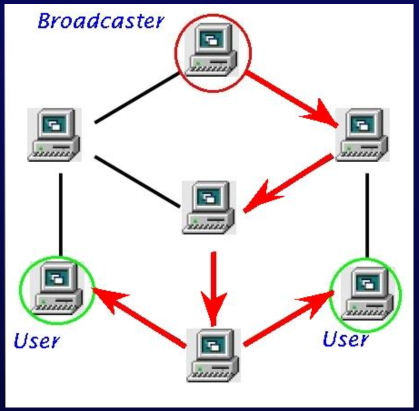
Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).



Packing Edge-disjoint Undirected (PEU):

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



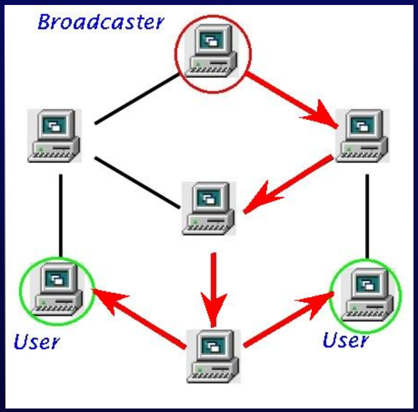
Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

Packing Edge-disjoint Undirected (PEU):

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

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

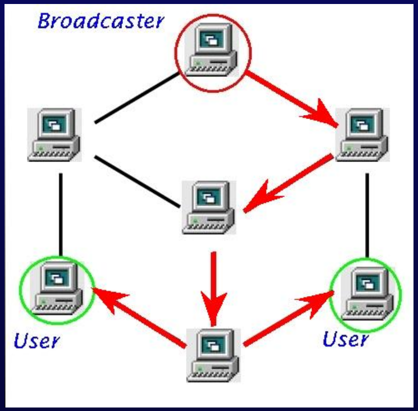
Packing Edge-disjoint Undirected (PEU):

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

Theorem (Nash-Williams & Tutte): If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

Packing Edge-disjoint Undirected (PEU):

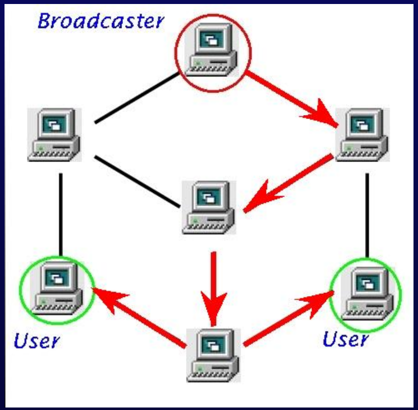
For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Theorem (Nash-Williams & Tutte): 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.

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

Packing Edge-disjoint Undirected (PEU):

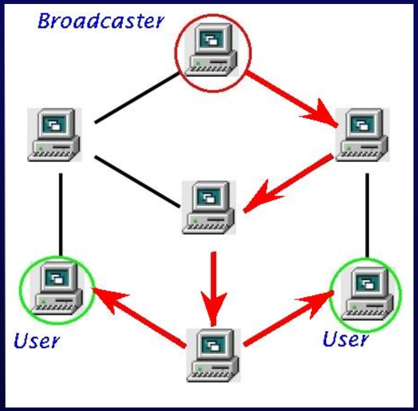
For $T \subseteq V$, T*-edge-connectivity* is the minimum number of edges whose removal disconnects two vertices of T.

Theorem (Nash-Williams & Tutte): 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 $k \geq 2$)

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

Packing Edge-disjoint Undirected (PEU):

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

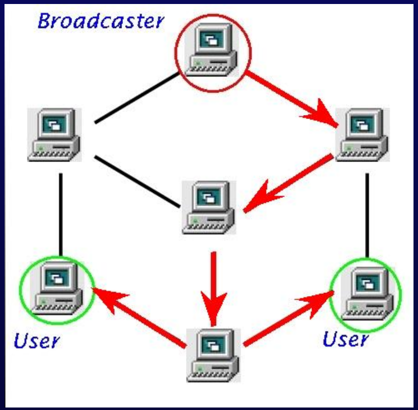
Theorem (Nash-Williams & Tutte): If G has V-edge-connectivity at least $2k$, then there are at least k edge-disjoint spanning trees in G.

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 [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.

To Graph Theory

Given a (directed/undirectied) graph $G(V, E)$ and a set of terminals $T \subseteq V$. Vertices in $V - T$ are called Steiner nodes. Find max number of edge-disjoint Steiner trees.



Observation: All leaves in a Steiner tree are terminals (otherwise, remove it).

Packing Edge-disjoint Undirected (PEU):

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

Theorem (Nash-Williams & Tutte): 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 [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 yields a polynomial time algorithm.

Results on PEU (cont'd)...

Theorem (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 and this is sharp.

Results on PEU (cont'd)...

Theorem (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 and this is sharp.

Theorem (Jain & Mahdian & S. '03): PEU is APX-hard.

Results on PEU (cont'd)...

Theorem (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 and this is sharp.

Theorem (Jain & Mahdian & S. '03): PEU is APX-hard.

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

Theorem (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 and this is sharp.

Theorem (Jain & Mahdian & S. '03): PEU is APX-hard.

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

LP formulation and fractional packing

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

Theorem (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 and this is sharp.

Theorem (Jain & Mahdian & S. '03): PEU is APX-hard.

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

LP formulation and fractional packing

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

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

Theorem (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 and this is sharp.

Theorem (Jain & Mahdian & S. '03): PEU is APX-hard.

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

LP formulation and fractional packing

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

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

Fractional PEU is the corresponding LP. The separation oracle for the dual LP is the min. Steiner Tree problem.

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

Theorem (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 a 1.55-approx algorithm.

Theorem (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 a 1.55-approx algorithm.

This theorem holds in more general settings and we will use this later.

Our results:

By a reduction form a variation of SAT:
Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees (independently by Kaski'04).

Theorem (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 a 1.55-approx algorithm.

This theorem holds in more general settings and we will use this later.

Our results:

By a reduction form a variation of SAT:
Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees (independently by Kaski'04).

Theorem 1: PEU is APX-hard even with 4 termianls, i.e. there is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$.

Theorem (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 a 1.55-approx algorithm.

This theorem holds in more general settings and we will use this later.

Our results:

By a reduction form a variation of SAT:
Given G and $T \subseteq V$, it is NP-hard to decide if G has two edge-disjoint Steiner trees (independently by Kaski'04).

Theorem 1: PEU is APX-hard even with 4 termianls, i.e. there is an absolute constant $c > 1$ s.t. there is no c-approximation algorithm for PEU even if $|T| = 4$, unless $P = NP$.

Proof idea:

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

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 2: 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 2: Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 3: PVU is APX-hard even if $|T|$ is constant.

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 2: Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 3: PVU is APX-hard even if $|T|$ is constant.

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 2: Given G and $T \subseteq V$, it is NP-hard to decide if G has two vertex-disjoint Steiner trees.

Theorem 3: PVU is APX-hard even if $|T|$ is constant.

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 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

Theorem 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

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

Theorem 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

Theorem 5: For any $\epsilon > 0$, there is an $O(m^{\frac{1}{2}+\epsilon})$-approximation for PED, with m being the number of edges.

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

Theorem 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

Theorem 5: 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 and relax it to LP (i.e. take fractional PED)

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

Theorem 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

Theorem 5: 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 and relax it to LP (i.e. take fractional PED)

2. Try to solve this LP (maybe approximately)

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

Theorem 4: There are poly. time approximation preserving reductions from PED to PVD and from PVD to PED.

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

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

Theorem 5: 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 and relax it to LP (i.e. take fractional PED)
2. Try to solve this LP (maybe approximately)
3. Use randomized rounding to obtain an integral solution.

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

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.

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.

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 $O(\log^{2-\epsilon} n)$ factor (Halperin & Krauthgamer '03). But . . .

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 $O(\log^{2-\epsilon} n)$ factor (Halperin & Krauthgamer '03). But . . .

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

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 $O(\log^{2-\epsilon} n)$ factor (Halperin & Krauthgamer '03). But . . .

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

The result of Jain, Mahdian, and Salavatipour can be extended to prove:

Theorem 6: There is an α-approx algorithm for fractional PED iff there is an α-approx algorithm for min directed Steiner tree.

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 $O(\log^{2-\epsilon} n)$ factor (Halperin & Krauthgamer '03). But . . .

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

The result of Jain, Mahdian, and Salavatipour can be extended to prove:

Theorem 6: 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.

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 $O(\log^{2-\epsilon} n)$ factor (Halperin & Krauthgamer '03). But . . .

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

The result of Jain, Mahdian, and Salavatipour can be extended to prove:

Theorem 6: 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.

Simple randomized rounding yields an $O(m^{\frac{1}{2}+\epsilon})$-approximation.

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 7: 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 7: Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We sketch the proof of a weaker version:

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

We sketch the proof of a weaker version:

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

Remark: The roof 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 7: Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We sketch the proof of a weaker version:

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

Remark: The roof 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 7: Unless $P = NP$, any approximation algorithm for PED has approximation factor $\Omega(m^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$.

We sketch the proof of a weaker version:

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

Remark: The roof 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?

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

We sketch the proof of a weaker version:

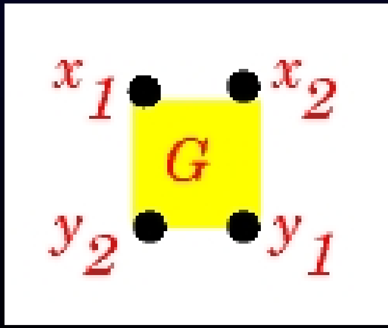
Theorem: Unless $P = NP$, any approx algorithm for PED has factor $\Omega(m^{\frac{1}{4}-\epsilon})$.

Remark: The roof 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?

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

We sketch the proof of a weaker version:

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

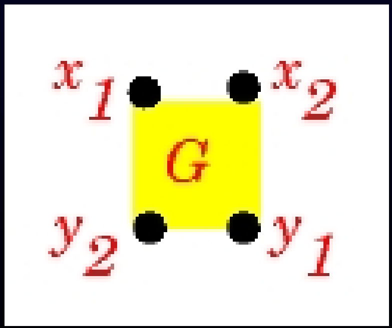
Remark: The roof does not rely on PCP theorem.

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?

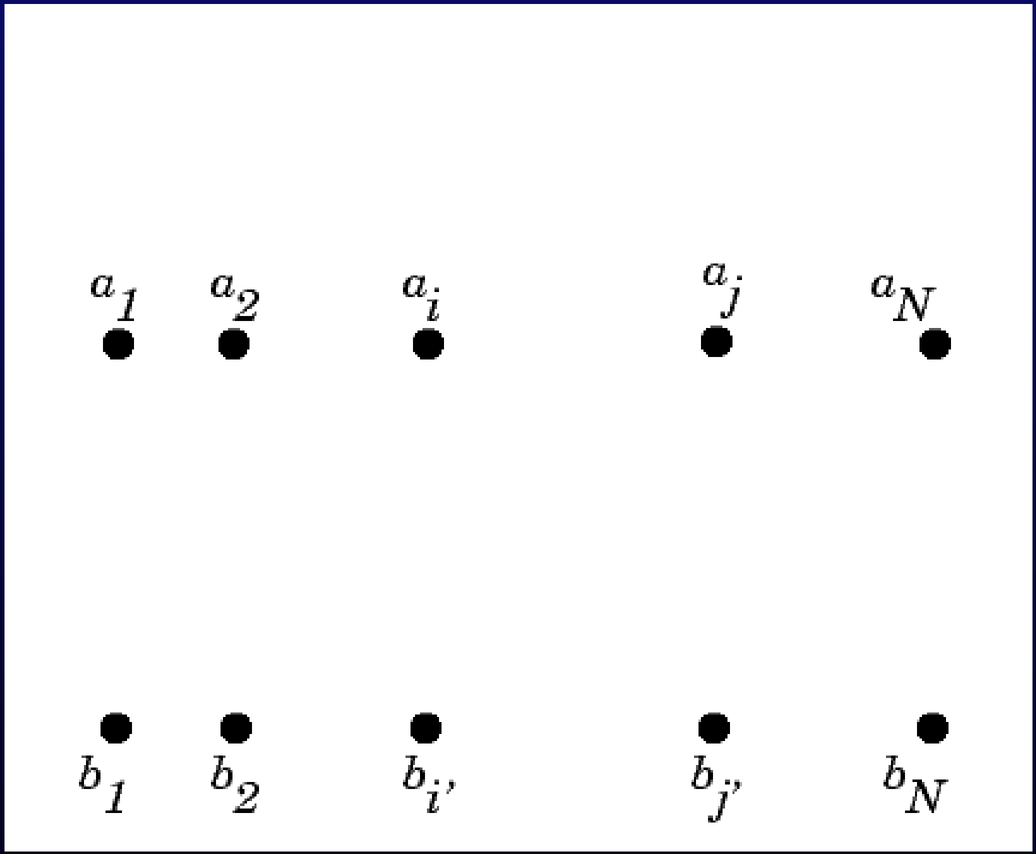
Let $I = (G, x_1, y_1, x_2, y_2)$ be an instance of 2DIRPATH and $\epsilon > 0$ be given.

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



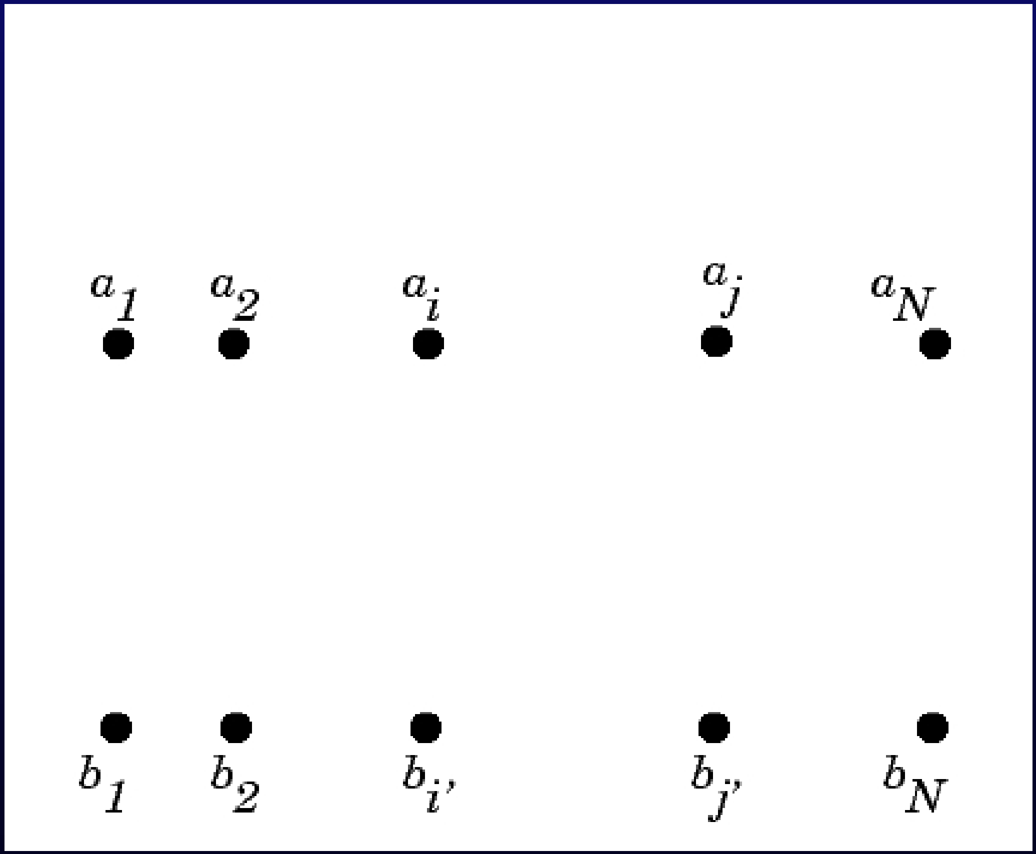
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

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



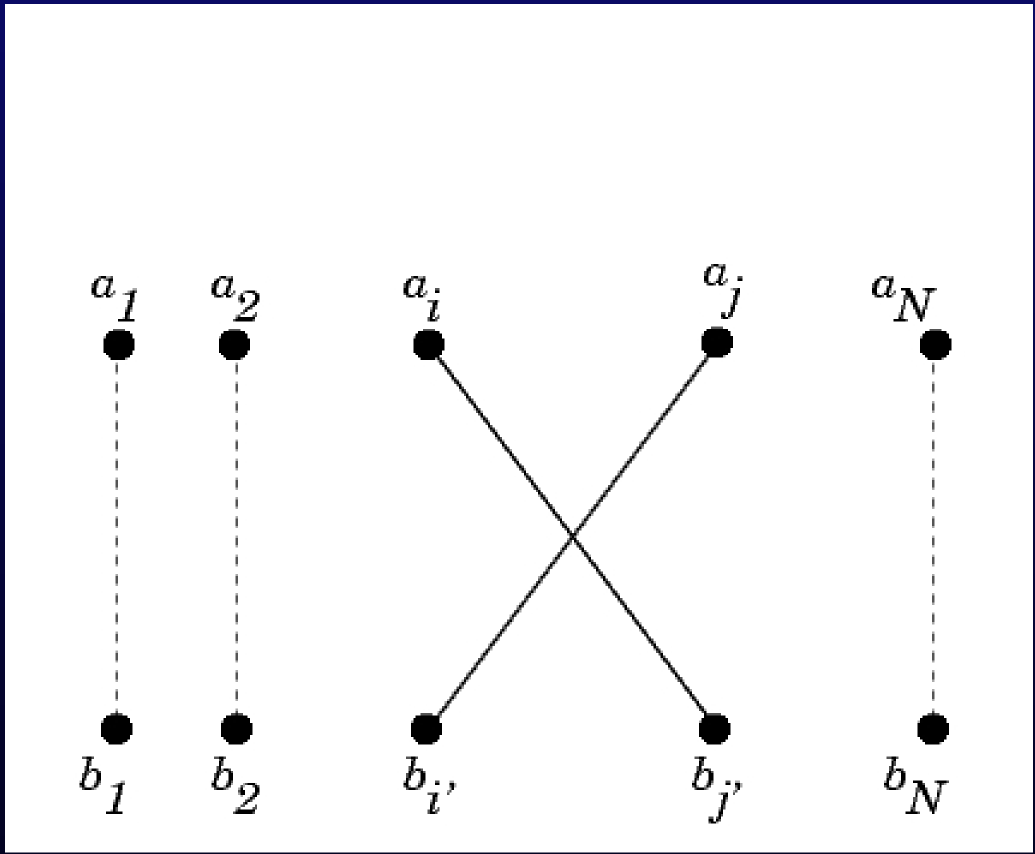
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

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 $B = \{b_1, \ldots, b_N\}$.

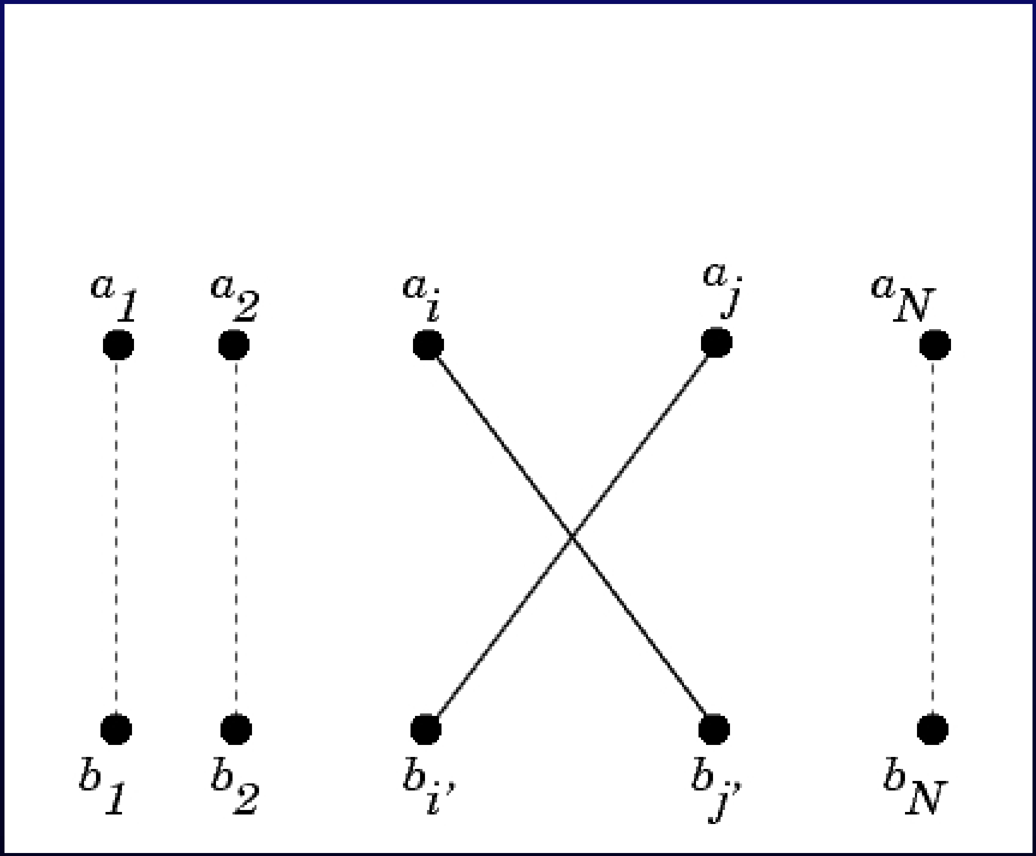
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.



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 $1 \leq i \neq j \leq N$.

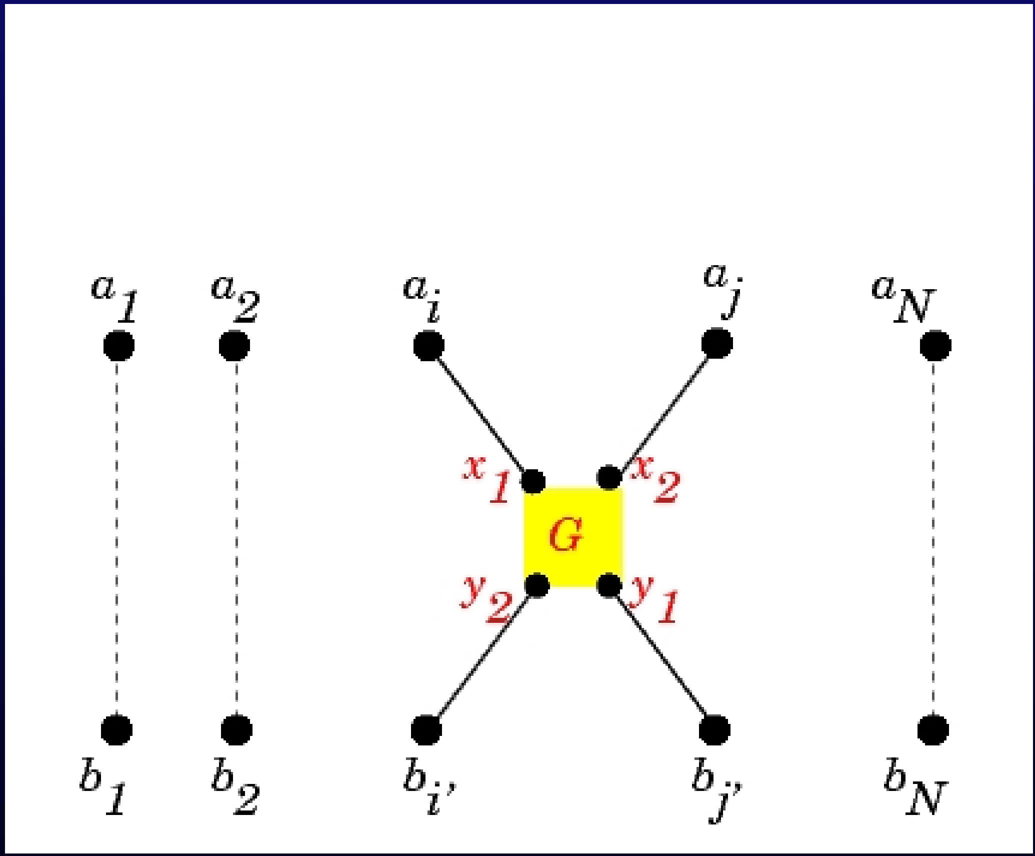
At each intersection put a copy of G.



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.

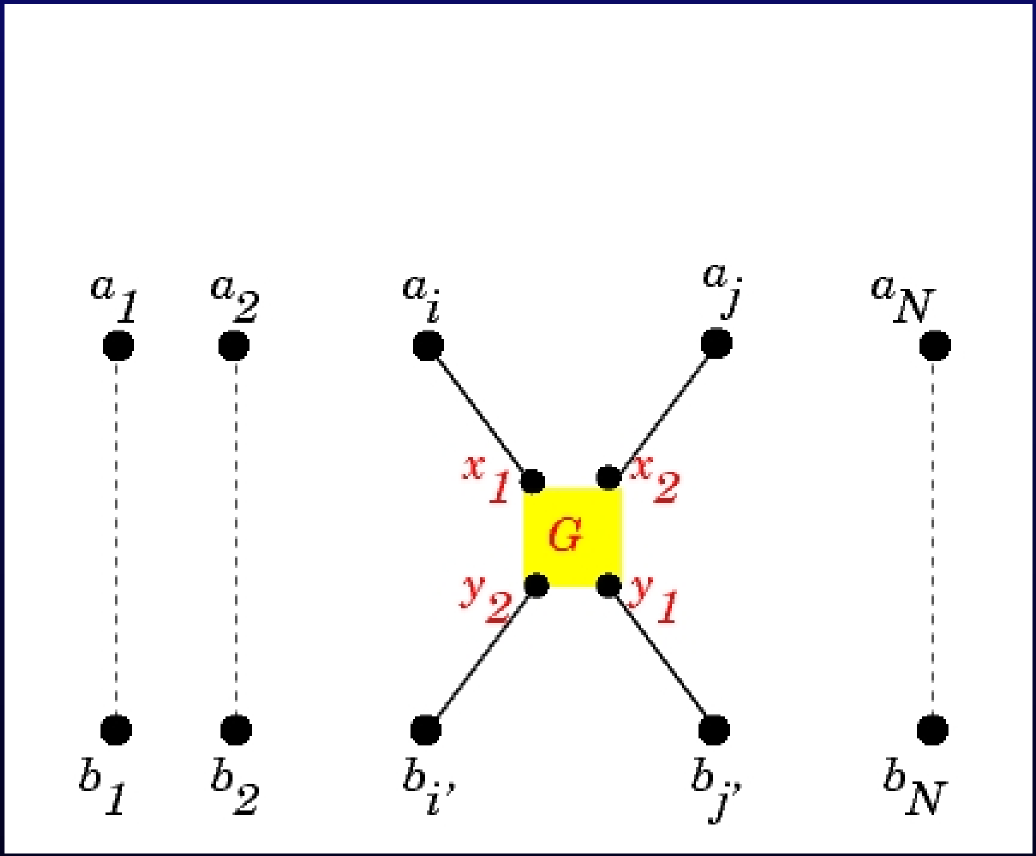


With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

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 $1 \leq i \leq N$

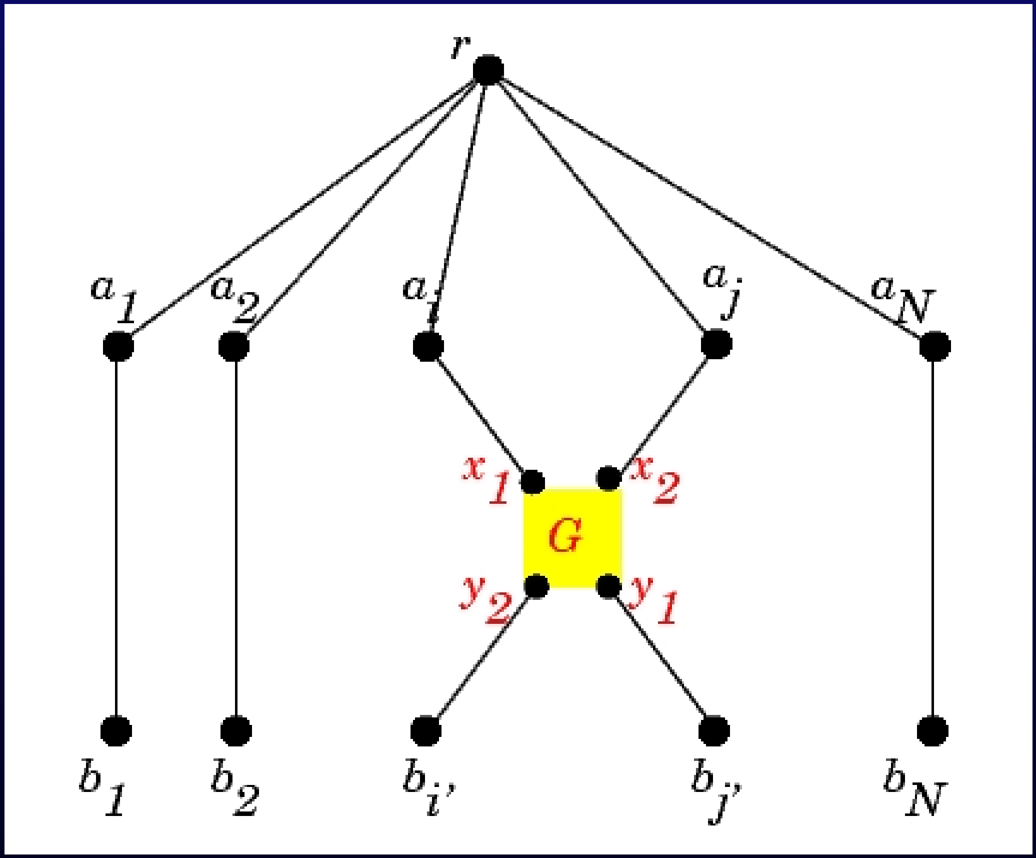


With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

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 $1 \leq i \leq N$



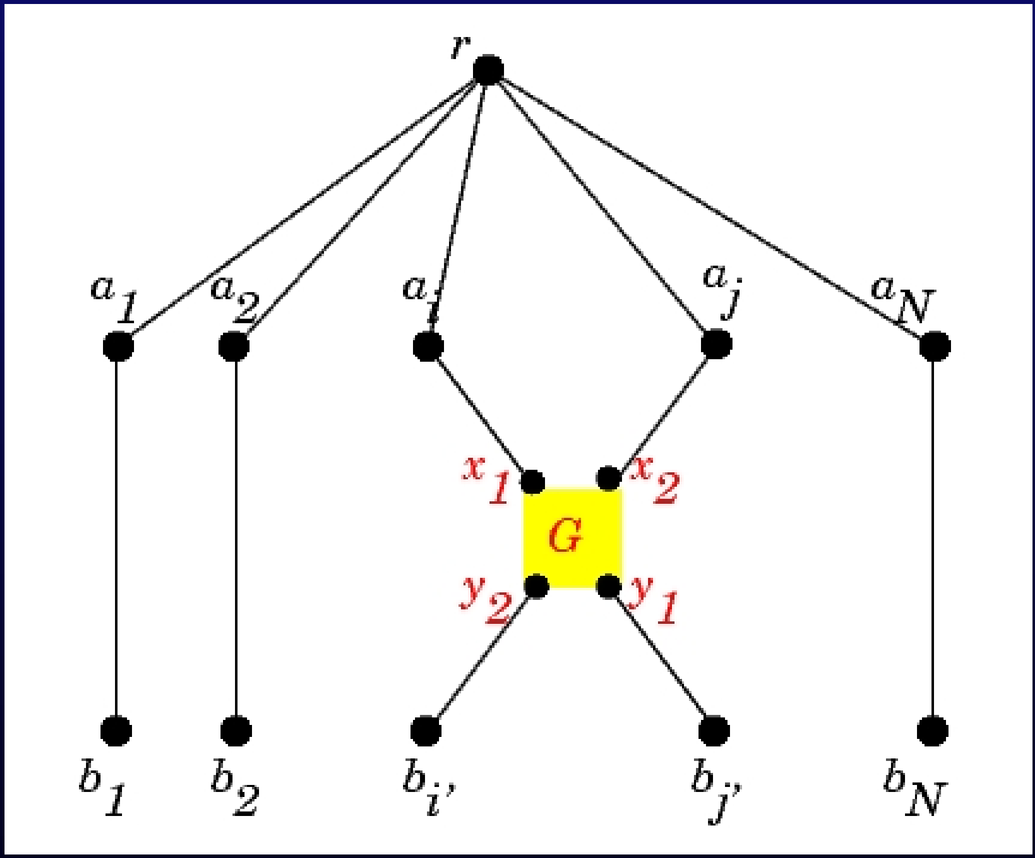
With $N = |E(G)|^{\frac{1}{\epsilon}}$, create two sets of vertices $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$.

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 $B = \{b_1, \ldots, b_N\}$.

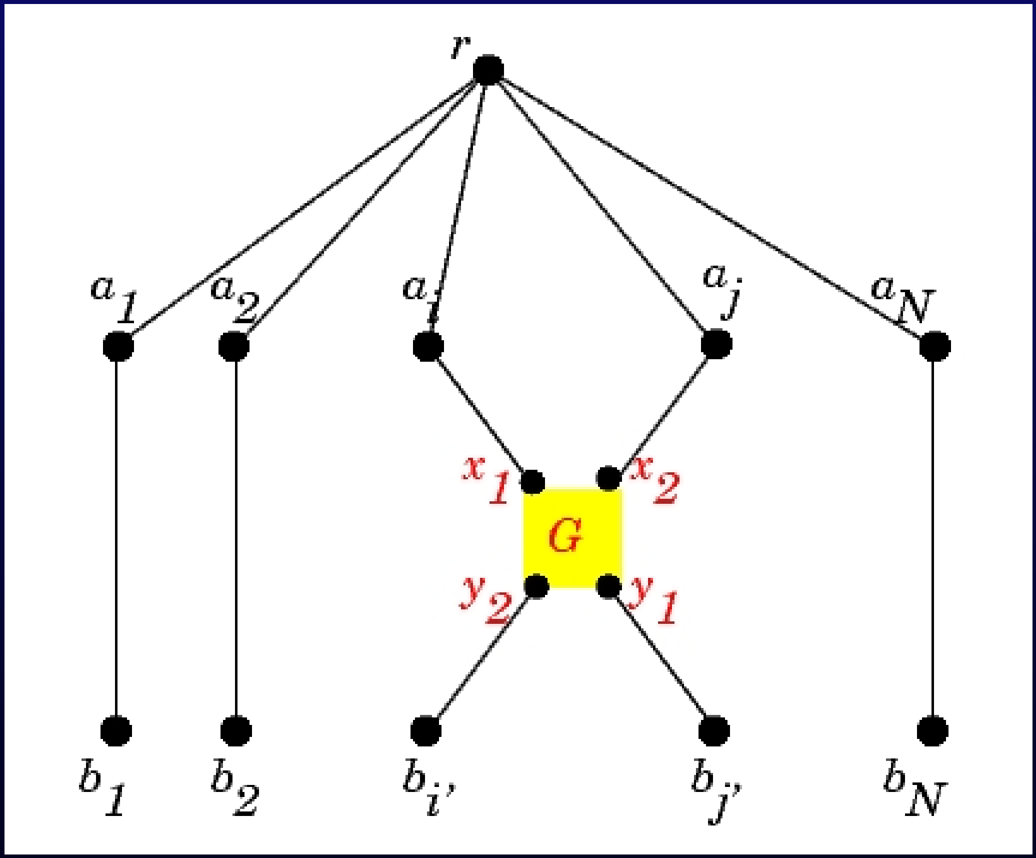
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

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 $B = \{b_1, \ldots, b_N\}$.

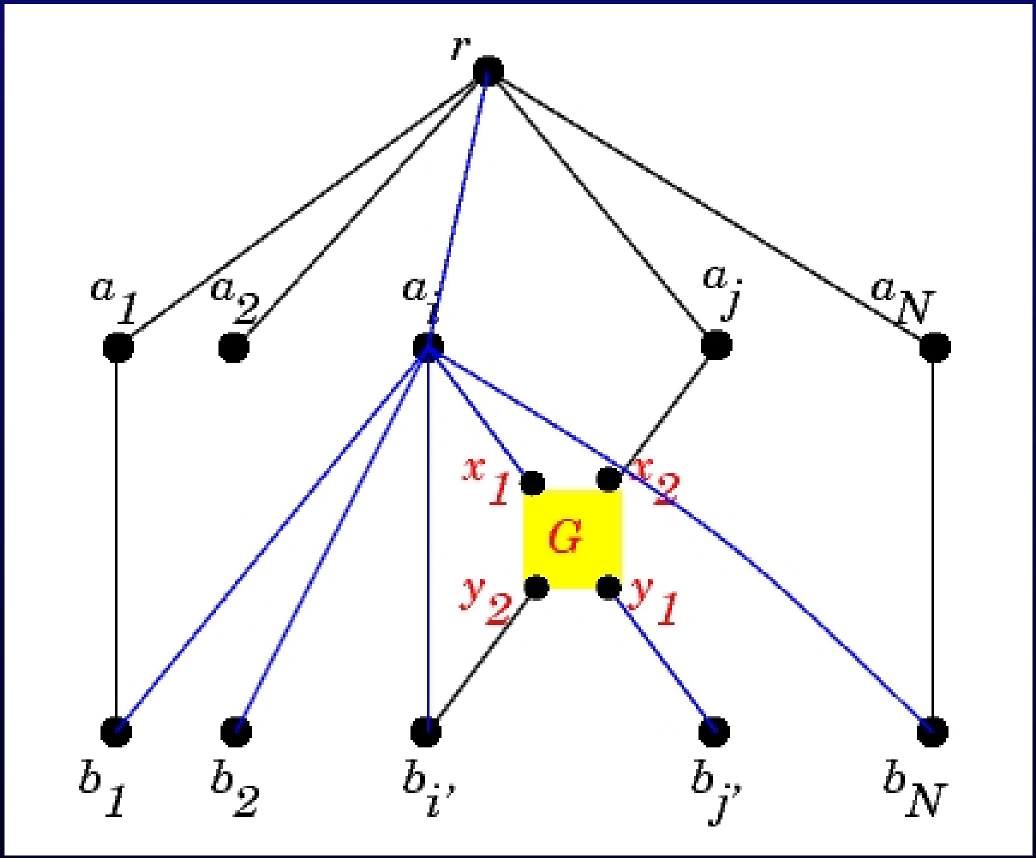
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

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 $B = \{b_1, \ldots, b_N\}$.

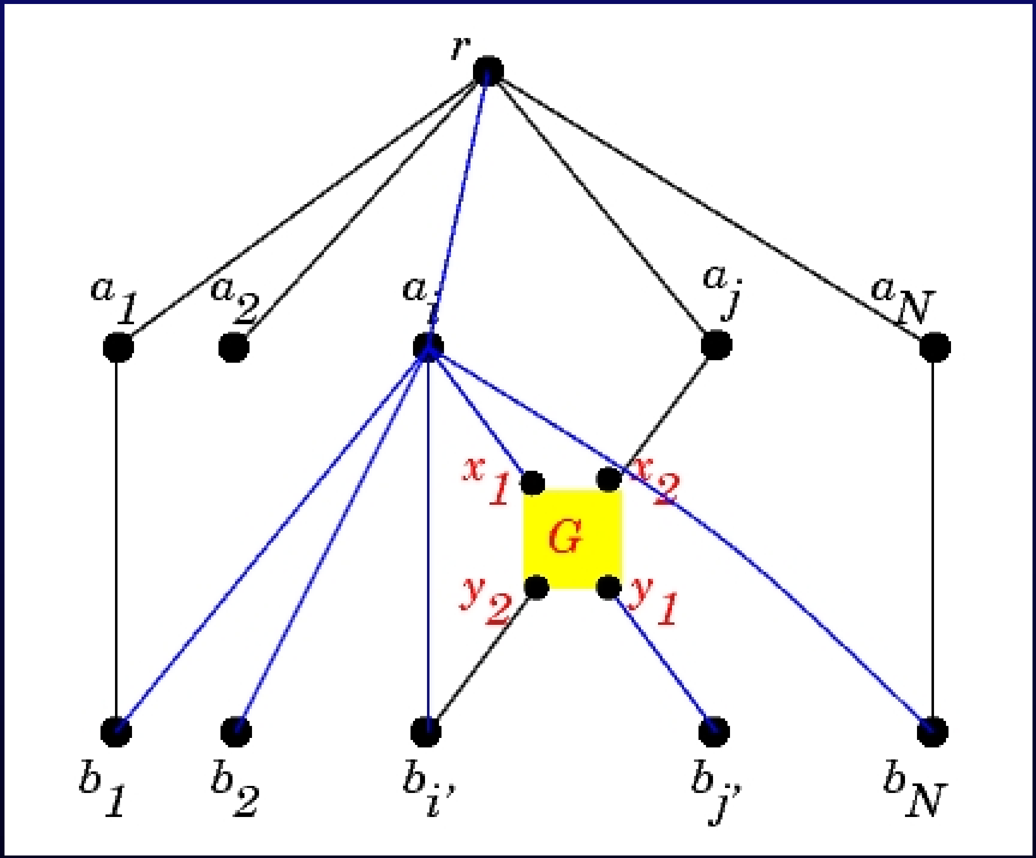
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

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 $B = \{b_1, \ldots, b_N\}$.

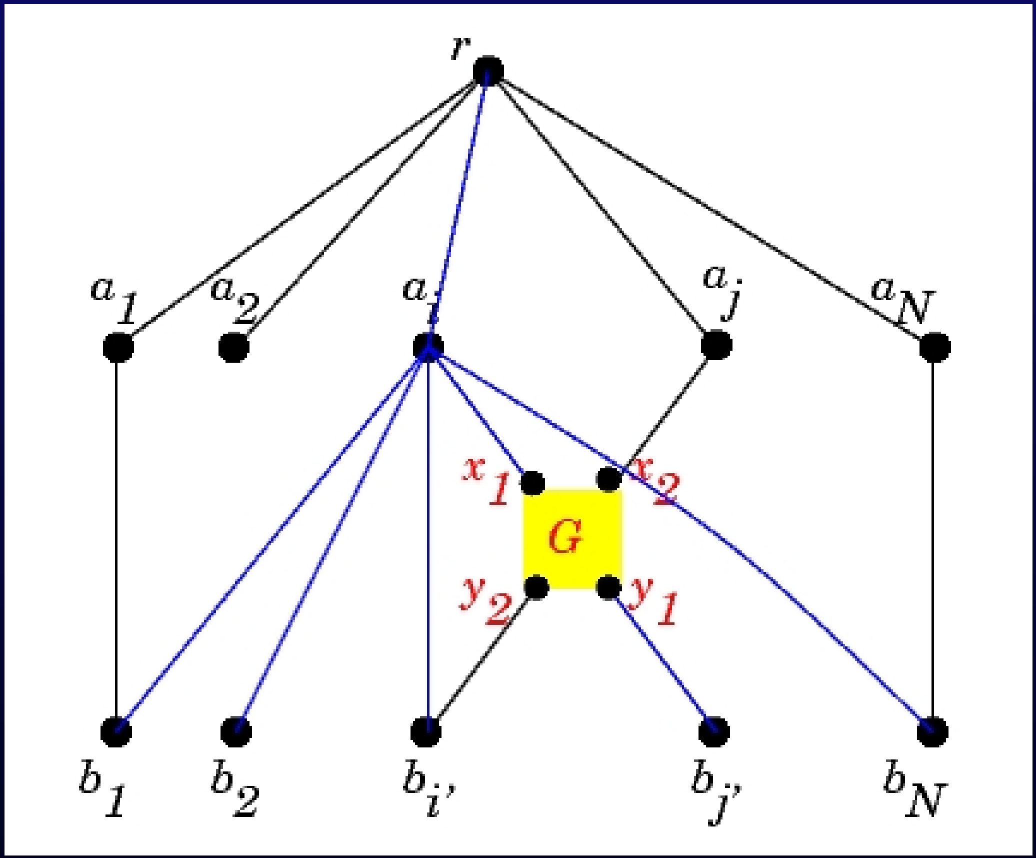
Create $a_i b_j$, for all $1 \leq i \neq j \leq N$.

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.

Using a similar reduction:

Theorem 8: It is NP-hard to approximate PVD within $\Omega(n^{\frac{1}{3}-\epsilon})$ factor.

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.

Using a similar reduction:

Theorem 8: It is NP-hard to approximate PVD within $\Omega(n^{\frac{1}{3}-\epsilon})$ factor.

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

Theorem 9: There is an $O(n^{\frac{1}{2}+\epsilon})$-approximation for PVD.

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.

Using a similar reduction:

Theorem 8: It is NP-hard to approximate PVD within $\Omega(n^{\frac{1}{3}-\epsilon})$ factor.

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

Theorem 9: There is an $O(n^{\frac{1}{2}+\epsilon})$-approximation for PVD.

Back to the undirected setting . . .

We showed that PEU and PVU are APX-hard for constnat $|T|$ and Lau gave 26-approximation for PEU. What about PVU?

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.

Using a similar reduction:

Theorem 8: It is NP-hard to approximate PVD within $\Omega(n^{\frac{1}{3}-\epsilon})$ factor.

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

Theorem 9: There is an $O(n^{\frac{1}{2}+\epsilon})$-approximation for PVD.

Back to the undirected setting . . .

We showed that PEU and PVU are APX-hard for constnat $|T|$ and Lau gave 26-approximation for PEU. What about PVU?

Theorem 10: PVU cannot be approximated with ratio $(1-\epsilon)\ln n$, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\log\log n})$.

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.

Using a similar reduction:

Theorem 8: It is NP-hard to approximate PVD within $\Omega(n^{\frac{1}{3}-\epsilon})$ factor.

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

Theorem 9: There is an $O(n^{\frac{1}{2}+\epsilon})$-approximation for PVD.

Back to the undirected setting . . .

We showed that PEU and PVU are APX-hard for constnat $|T|$ and Lau gave 26-approximation for PEU. What about PVU?

Theorem 10: PVU cannot be approximated with ratio $(1-\epsilon)\ln n$, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\log\log n})$.

Theorem 11: We can approximate PVU within $O(\log n\sqrt{n})$.

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$ for $\|T\| = 4$
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)	$1+\epsilon_0$ for $\lvert T\rvert = 4$
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)	$1+\epsilon_0$ for $\|T\|=4$
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)	$1+\epsilon_0$ for $\|T\| = 4$
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?
- We have found an $O(\log^2 n)$ approx for PVU.

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$ for $\lvert T\rvert = 4$
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?
- We have found an $O(\log^2 n)$ approx for PVU.
  Is there an $O(\log n)$ approx?

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1 + \epsilon_0$ for $\|T\| = 4$
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?
- We have found an $O(\log^2 n)$ approx for PVU.
  Is there an $O(\log n)$ approx?
- What is the integrality gap for PVU?

Summary of results and Open problems

Problems	Approx. Alg	Hardness
PEU	26 (L. Lau)	$1+\epsilon_0$ for $\lvert T\rvert = 4$
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?
- We have found an $O(\log^2 n)$ approx for PVU.
  Is there an $O(\log n)$ approx?
- What is the integrality gap for PVU?

Thanks!