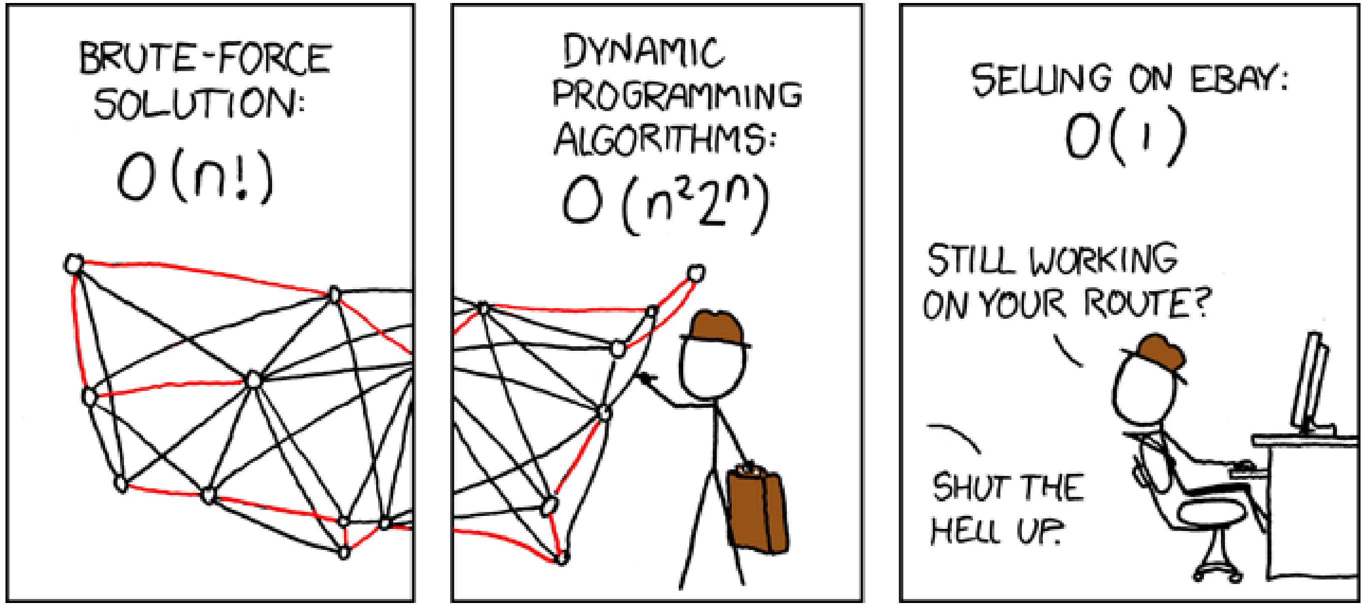
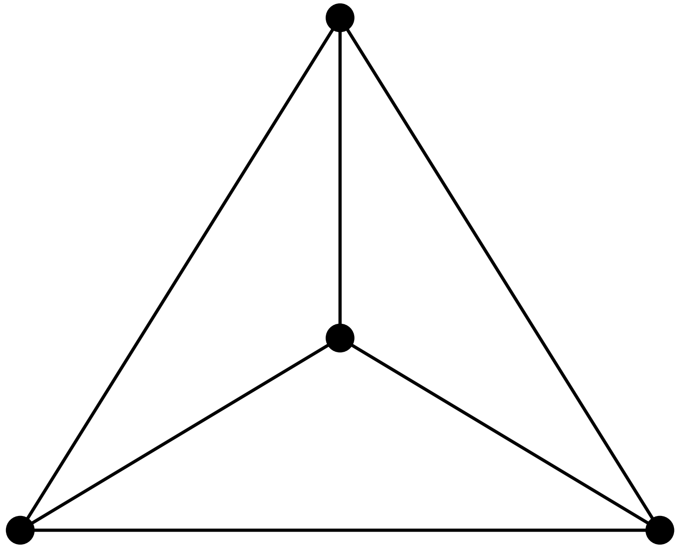
Approximation Algorithms for Steiner Tree, TSP and Multiway Cut



https://xkcd.com/399/

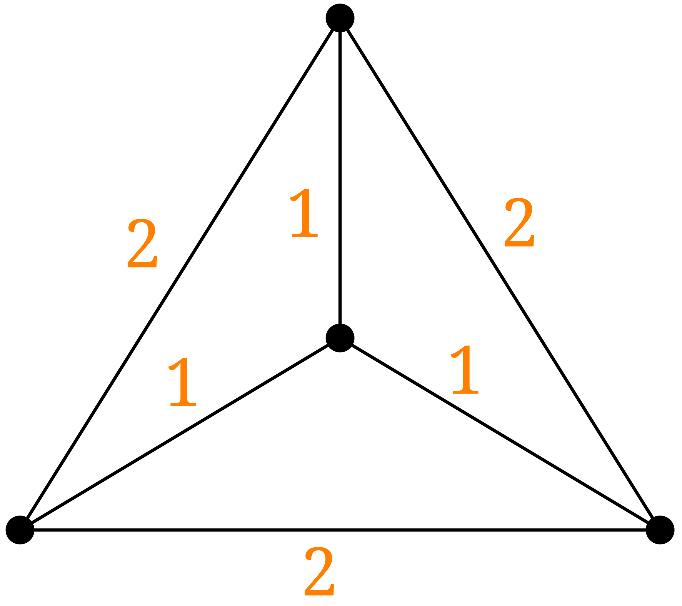
SteinerTree

Given: A graph G



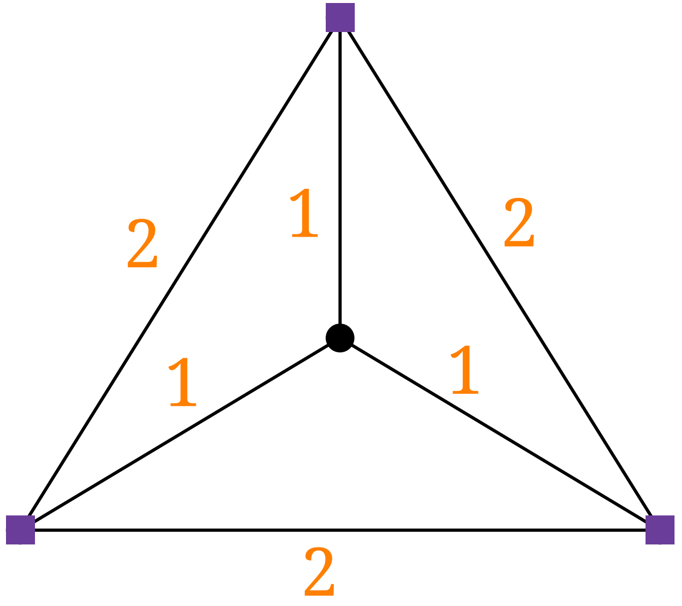
SteinerTree

Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$



STEINERTREE

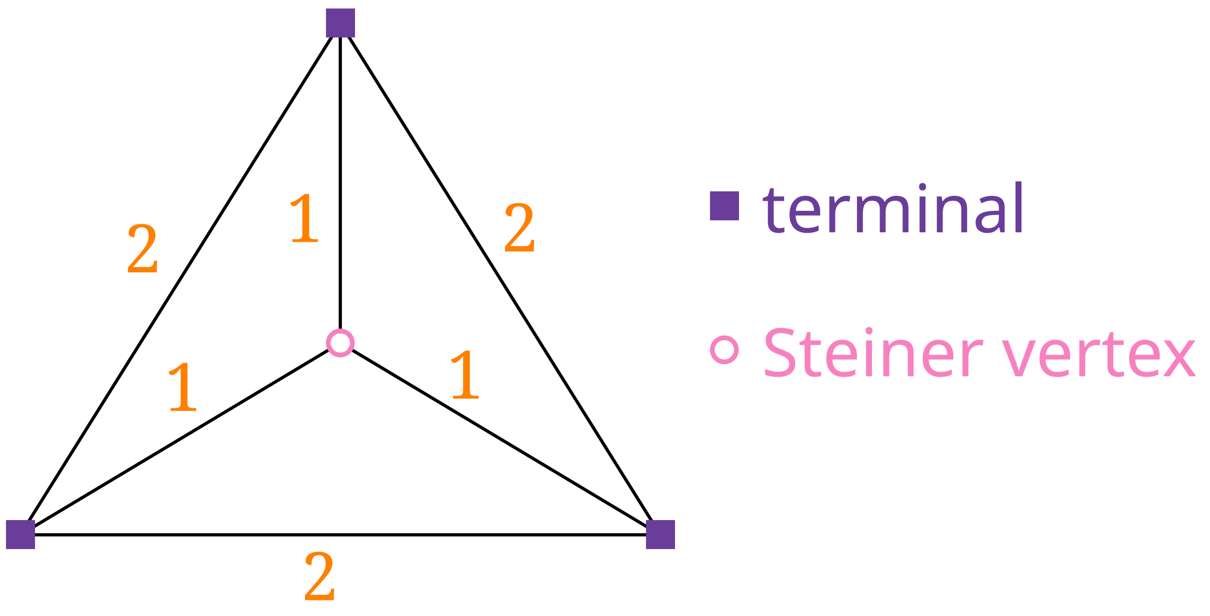
Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of terminals



■ terminal

SteinerTree

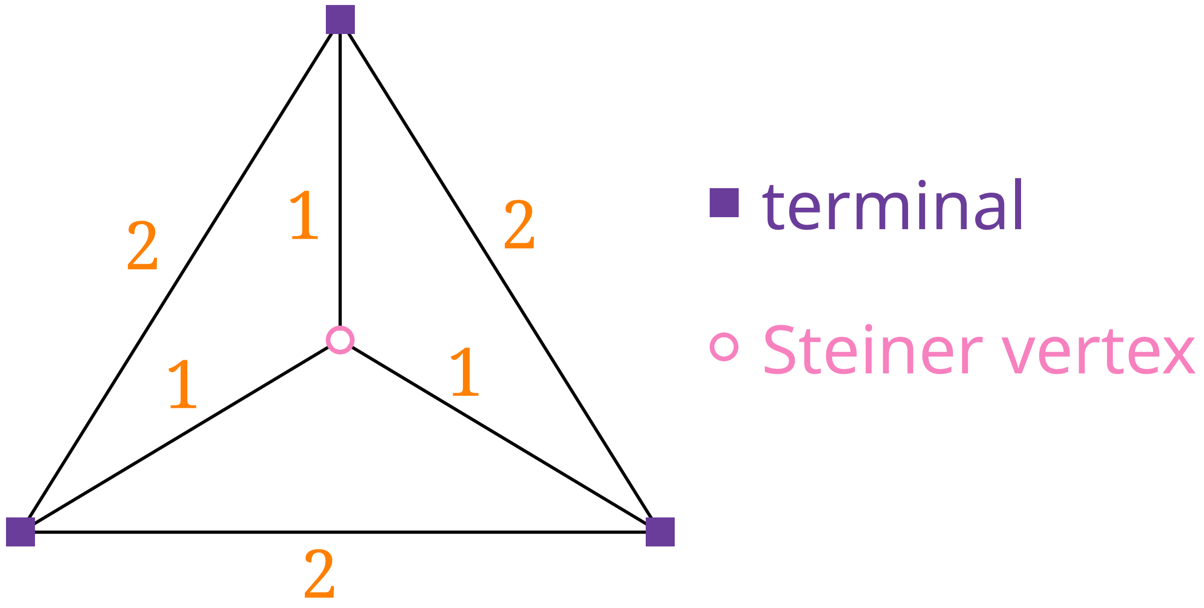
Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.



SteinerTree

Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.

Find: A subtree B of G that
- contains all terminals (i.e., $T \subseteq V(B)$) and

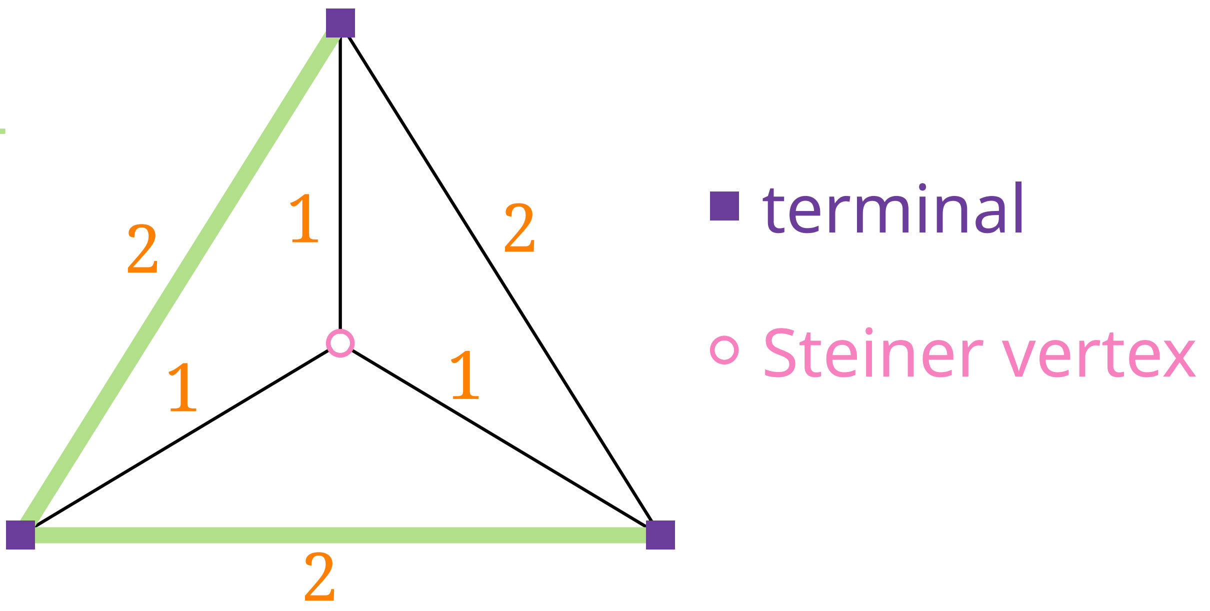


STEINERTREE

Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^{+}$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.

Find: A subtree B of G that
- contains all terminals (i.e., $T \subseteq V(B)$) and

valid solution with cost 4



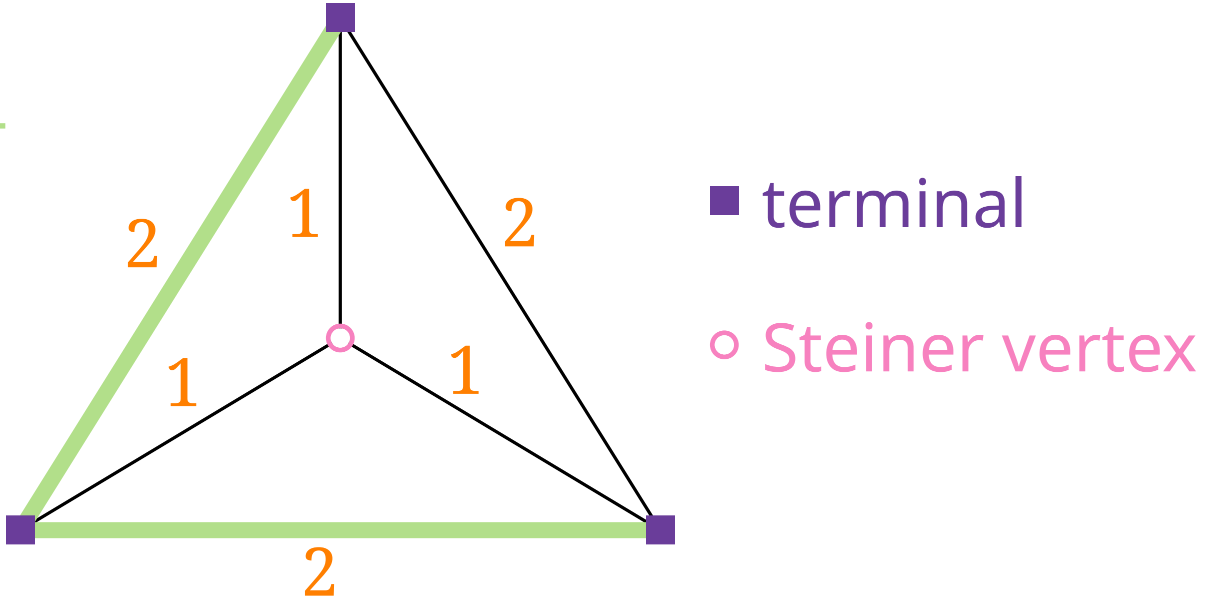
SteinerTree

Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.

Find: A subtree B of G that

- contains all terminals (i.e., $T \subseteq V(B)$) and
- has minimum cost $c(B) := \sum_{e \in E(B)} c(e)$ among all subtrees with this property.

valid solution with cost 4



STEINERTREE

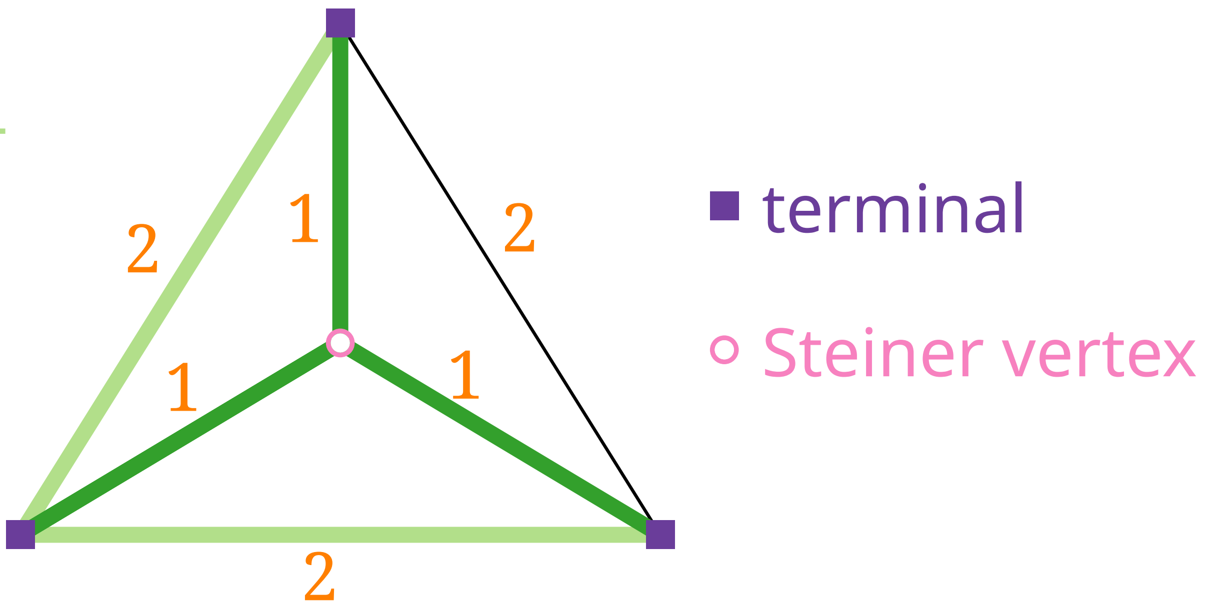
Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.

Find: A subtree B of G that

- contains all terminals (i.e., $T \subseteq V(B)$) and
- has minimum cost $c(B) := \sum_{e \in E(B)} c(e)$ among all subtrees with this property.

valid solution with cost 4

optimum solution
with cost 3



SteinerTree

Given: A graph G with edge weights $c\colon E(G) \to \mathbb{Q}^{+}$
and a partition of $V(G)$ into a set T of terminals
and a set S of Steiner vertices.

Find: A subtree B of G that

- contains all terminals (i.e., $T \subseteq V(B)$) and
- has minimum cost $c(B) := \sum_{e \in E(B)} c(e)$ among all subtrees with this property.

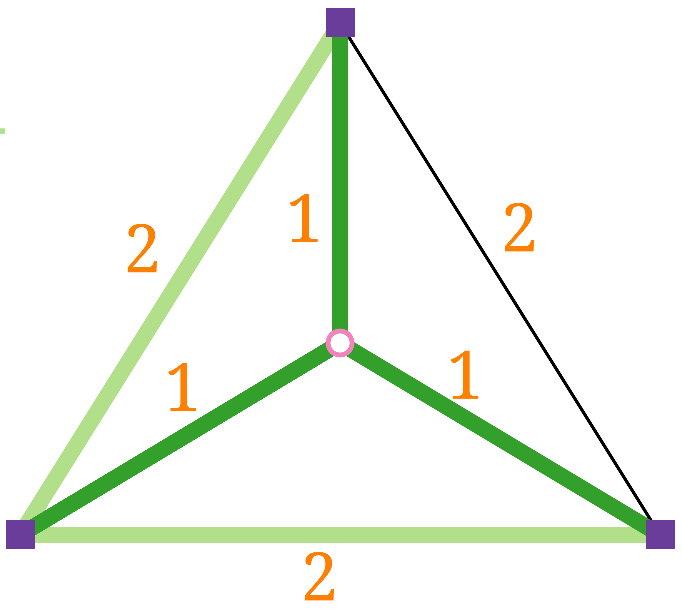
Special cases:

$T = V$: MST in P

$T = \{s, t\}$: shortestPath in P

valid solution with cost 4

optimum solution with cost 3



■ terminal

○ Steiner vertex

MetricSteinerTree

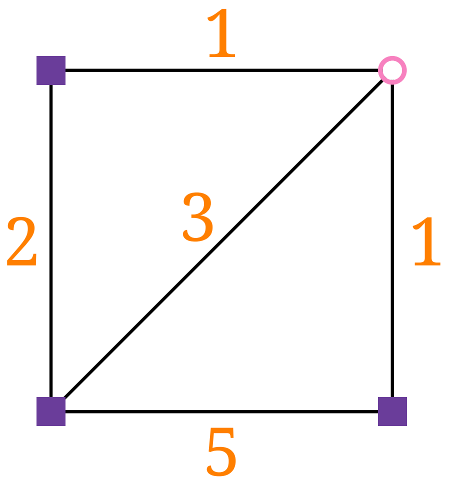
Restriction of SteinerTree where the graph G is complete and the cost function is metric,

MetricSteinerTree

Restriction of SteinerTree where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.

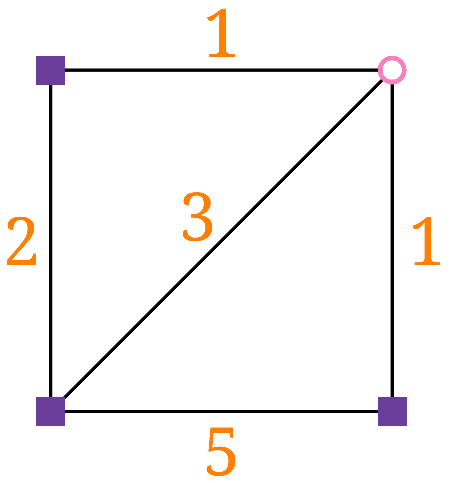
MetricSteinerTree

Restriction of SteinerTree where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



MetricSteinerTree

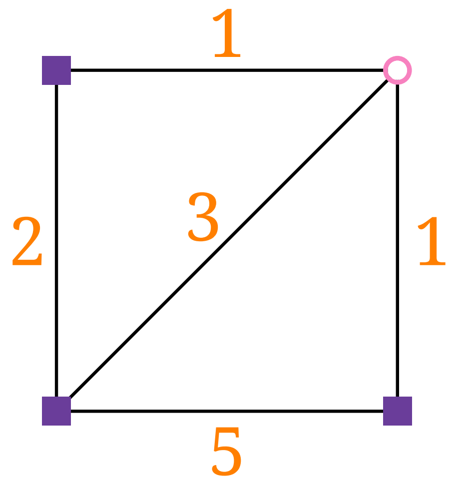
Restriction of SteinerTree where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



not complete

MetricSteinerTree

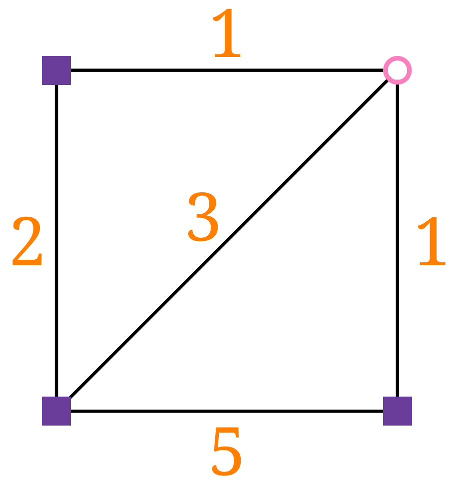
Restriction of SteinerTree where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u,w) \leq c(u,v) + c(v,w)$.



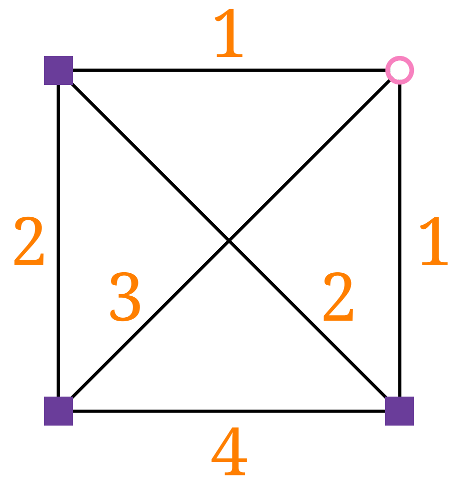
not complete
not metric

MetricSteinerTree

Restriction of SteinerTree where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u,w) \leq c(u,v) + c(v,w)$.

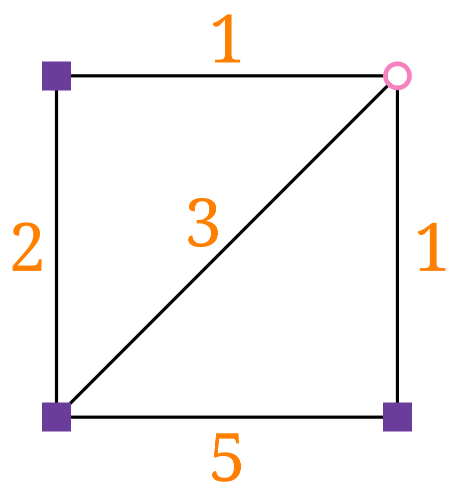


not complete
not metric

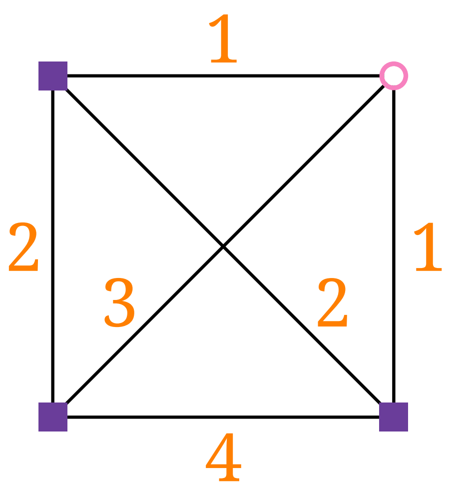


METRICSTEINERTREE

Restriction of STEINERTREE where the graph G is complete and the cost function is metric, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



not complete
not metric



complete
metric

Approximation-Preserving Reductions

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems.

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems.

problems Π_1 Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

problems	Π_1	Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.

problems	Π_1	Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.

problems	Π_1	Π_2
instances	I_1	

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.

$$\begin{array}{lccc} \text{problems} & \Pi_1 & & \Pi_2 \\ \text{instances} & I_1 & \xrightarrow{\quad f \quad} & I_2 \end{array}$$

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.
- For each feasible solution t of I_2,

  $s = g(I_1, t)$ is a feasible sol. of I_1 with $\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t)$.

problems	Π_1	Π_2
instances	$I_1 \xrightarrow{f}$	I_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,

  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.
- For each feasible solution t of I_2,

  $s = g(I_1, t)$ is a feasible sol. of I_1 with $\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t)$.

$$
\begin{tikzcd}
\text{problems} & \Pi_1 & \Pi_2 \\
\text{instances} & I_1 \arrow[r, "f"] & I_2 \arrow[d] \\
\text{solutions} & & t
\end{tikzcd}
$$

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

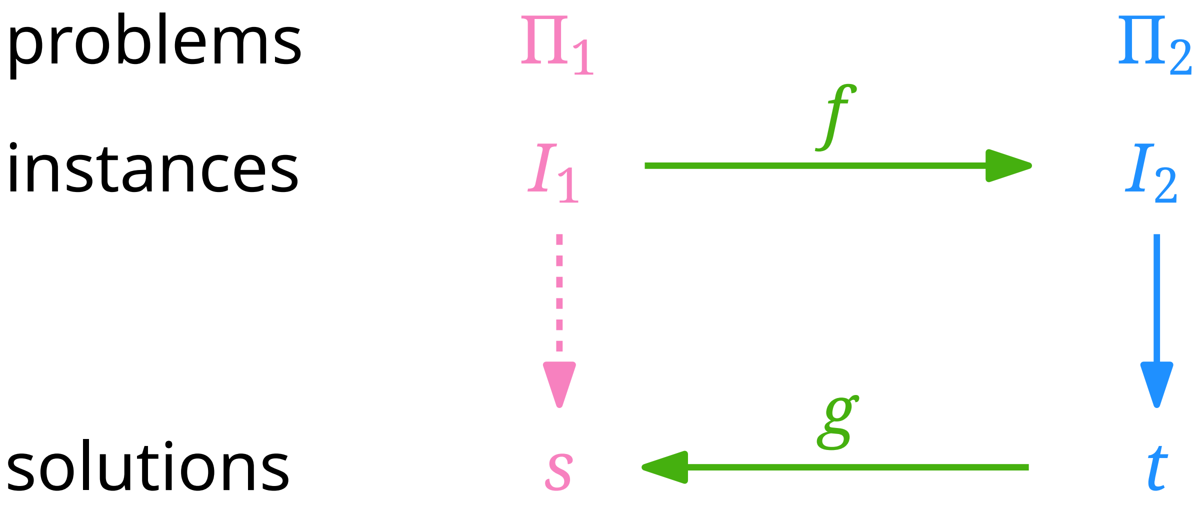
- For each instance I_1 of Π_1,
  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.
- For each feasible solution t of I_2,
  $s = g(I_1, t)$ is a feasible sol. of I_1 with $\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t)$.

$$\begin{tikzcd}
\text{problems} & \Pi_1 & & \Pi_2 \\
\text{instances} & I_1 \arrow[rr, "f"] & & I_2 \arrow[d] \\
\text{solutions} & s & & t \arrow[ll, "g"']
\end{tikzcd}$$

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An approximation- preserving reduction from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1,
  $I_2 = f(I_1)$ is an instance of Π_2 with $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$.
- For each feasible solution t of I_2,
  $s = g(I_1, t)$ is a feasible sol. of I_1 with $\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t)$.



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2.

$$
\begin{tikzcd}
\text{problems} & \Pi_1 & \Pi_2 \\
\text{instances} & I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d] \\
\text{solutions} & s & t \arrow[l, "g"']
\end{tikzcd}
$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-? approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

$$\begin{tikzcd}
\text{problems} & \Pi_1 & & \Pi_2 \\
\text{instances} & I_1 \arrow[rr, "f"] \arrow[dd, dashed] & & I_2 \arrow[dd] \\
& & & \\
\text{solutions} & s & & t \arrow[ll, "g"']
\end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-? approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

$$
\begin{tikzcd}
\text{problems} & \Pi_1 & & \Pi_2 \\
\text{instances} & I_1 \arrow[rr, "f"] \arrow[d, dashed] & & I_2 \arrow[d, "\alpha\text{-approximation}"] \\
\text{solutions} & s & & t \arrow[ll, "g"]
\end{tikzcd}
$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

$$\begin{tikzcd}
\text{problems} & \Pi_1 & & \Pi_2 \\
\text{instances} & I_1 \arrow[rr, "f"] \arrow[dd, dashed, "\alpha\text{-approximation}"'] & & I_2 \arrow[dd, "\alpha\text{-approximation}"] \\
 & & & \\
\text{solutions} & s & & t \arrow[ll, "g"']
\end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

$$\begin{tikzcd}
\Pi_1 & \Pi_2 \\
I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d] \\
s & t \arrow[l, "g"]
\end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

$$\begin{tikzcd} \Pi_1 & \Pi_2 \\ I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d] \\ s & t \arrow[l, "g"] \end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 :=$ $t :=$ and $s :=$

$$\begin{tikzcd} \Pi_1 & \Pi_2 \\ I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d] \\ s & t \arrow[l, "g"] \end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t :=$ and $s :=$

$$\begin{tikzcd} \Pi_1 & \Pi_2 \\ I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d] \\ s & t \arrow[l, "g"] \end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s :=$

$$\begin{tikzcd}
\Pi_1 & \Pi_2 \\
I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d, "A"] \\
s & t \arrow[l, "g"]
\end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s := g(I_1, t)$.

$$\begin{tikzcd} \Pi_1 & \Pi_2 \\ I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d, "A"] \\ s & t \arrow[l, "g"'] \end{tikzcd}$$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s := g(I_1, t)$.

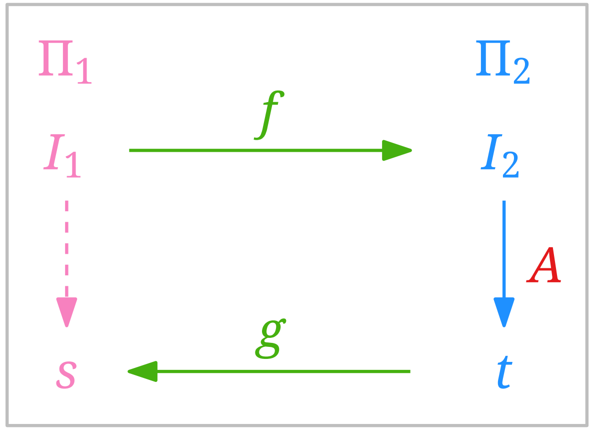
$$\begin{tikzcd}
\Pi_1 & \Pi_2 \\
I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d, "A"] \\
s & t \arrow[l, "g"]
\end{tikzcd}$$

Then:

$\mathrm{obj}_{\Pi_1}(I_1, s) \leq$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.



Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s := g(I_1, t)$.

Then:

$\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t) \leq$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

$$
\begin{tikzcd}
\Pi_1 & \Pi_2 \\
I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d, "A"] \\
s & t \arrow[l, "g"]
\end{tikzcd}
$$

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s := g(I_1, t)$.

Then:

$\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t) \leq \alpha \cdot \mathrm{OPT}_{\Pi_2}(I_2) \leq$

Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2. Then there is a factor-α approximation algorithm of Π_1 for each factor-α approximation algorithm of Π_2.

$$\begin{tikzcd}
\Pi_1 & \Pi_2 \\
I_1 \arrow[r, "f"] \arrow[d, dashed] & I_2 \arrow[d, "A"] \\
s & t \arrow[l, "g"']
\end{tikzcd}$$

Proof.

Let A be a factor-α approx. alg. for Π_2.

Let I_1 be an instance of Π_1.

Set $I_2 := f(I_1)$, $t := A(I_2)$ and $s := g(I_1, t)$.

Then:

$$\mathrm{obj}_{\Pi_1}(I_1, s) \le \mathrm{obj}_{\Pi_2}(I_2, t) \le \alpha \cdot \mathrm{OPT}_{\Pi_2}(I_2) \le \alpha \cdot \mathrm{OPT}_{\Pi_1}(I_1).$$

Reduction to MetricSteinerTree

MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot{\cup}\, S$

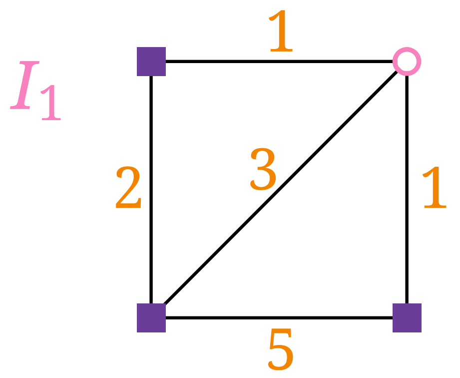
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\qquad I_1 \xrightarrow{f} I_2$

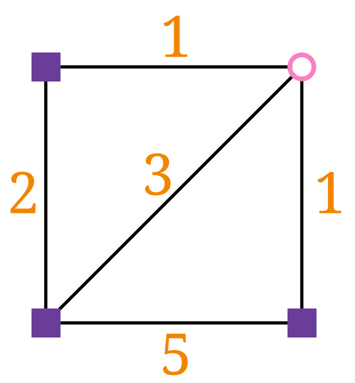
Instance I_1 of SteinerTree:

Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

I_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

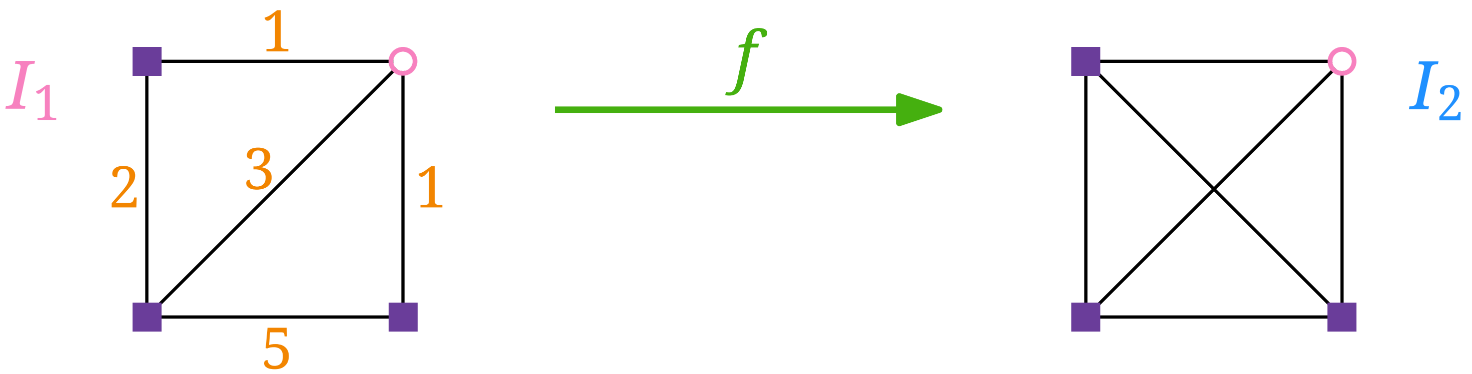
Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1



METRICSTEINERTREE

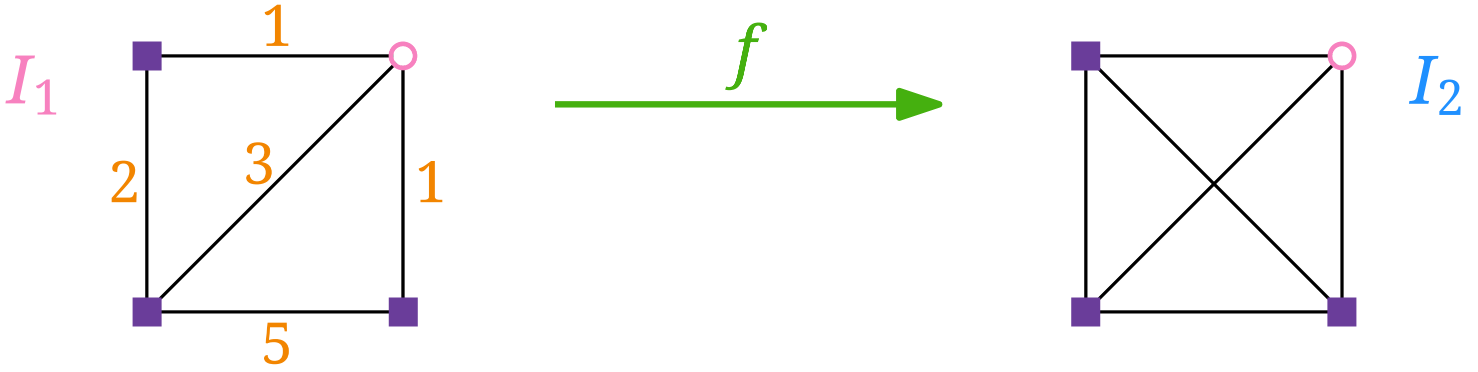
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:
Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1
$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

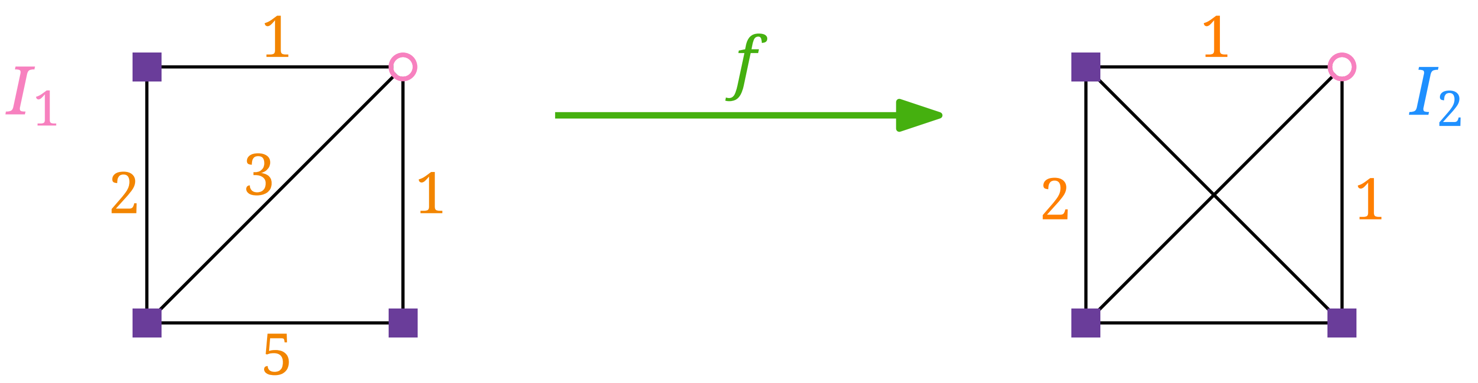
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \mathbin{\dot\cup} S$

Metric instance $I_2 := f(I_1)$:
Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1
$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

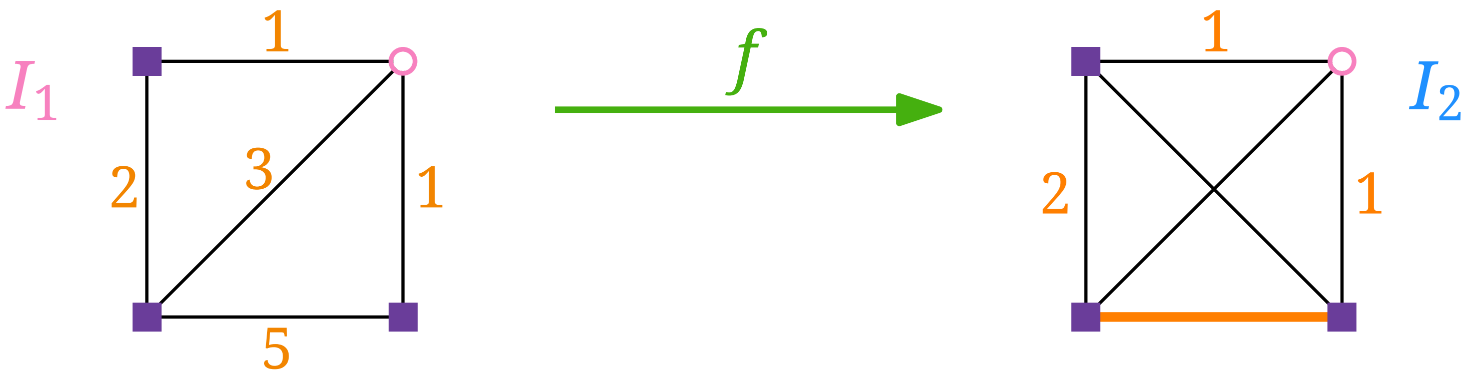
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \mathbin{\dot\cup} S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

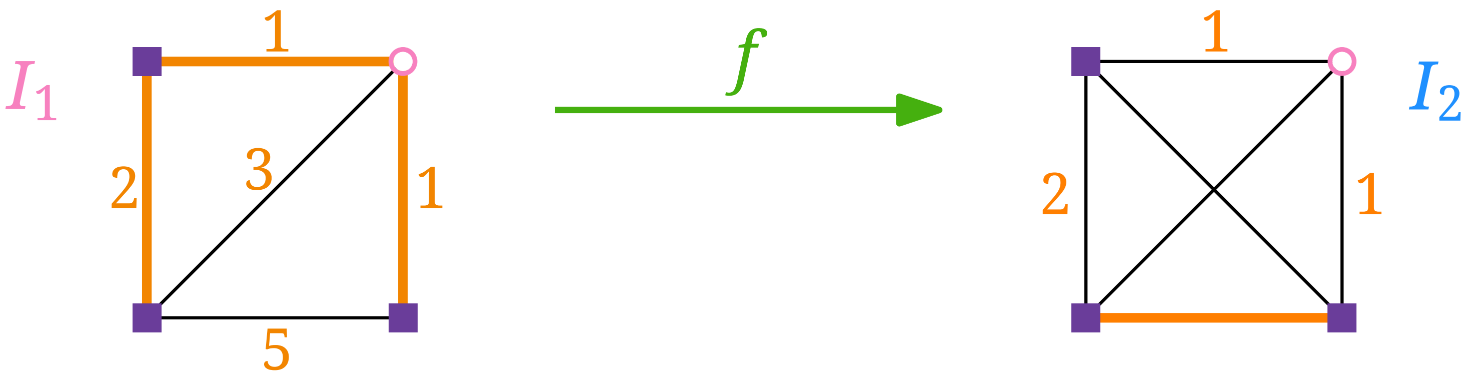
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

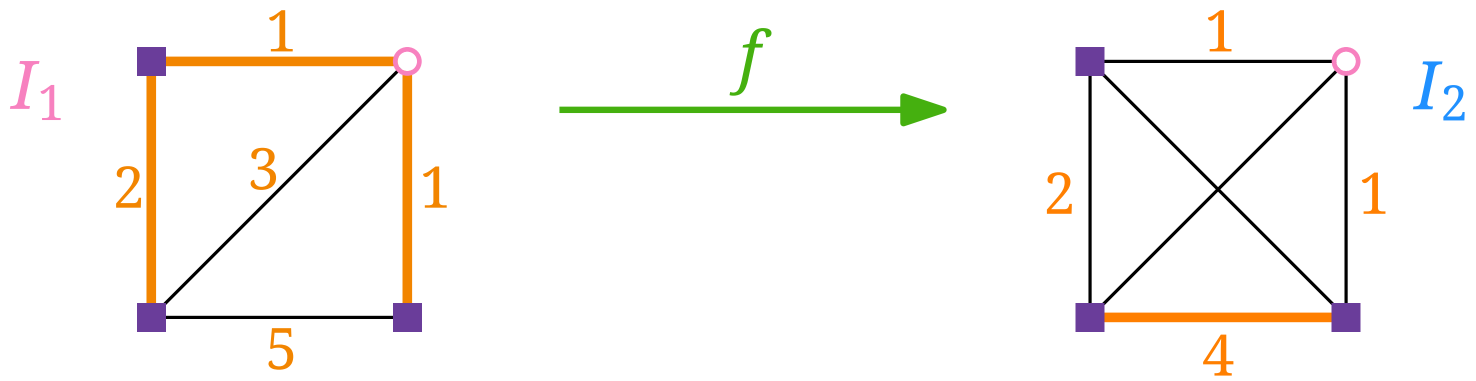
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

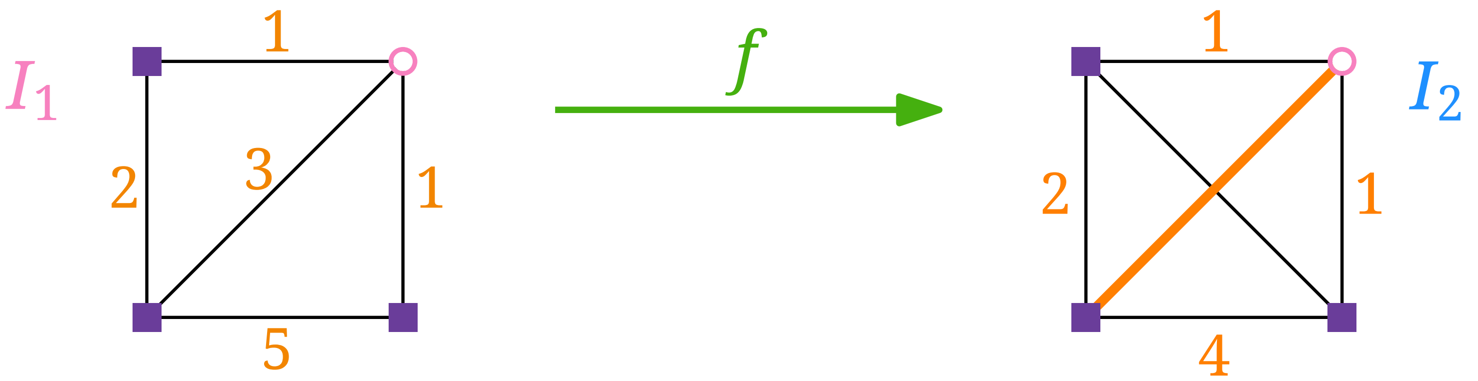
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

> **Theorem.** There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

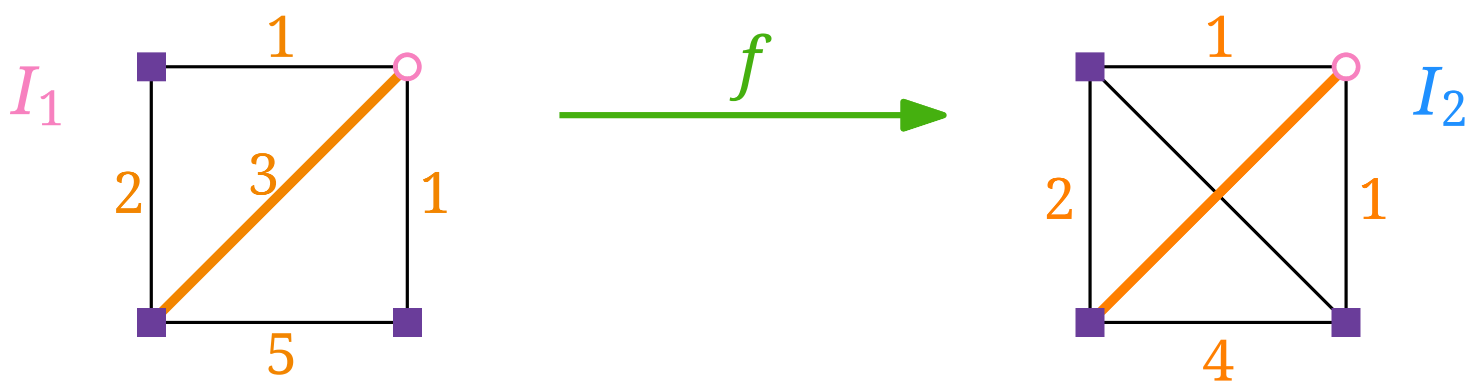
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

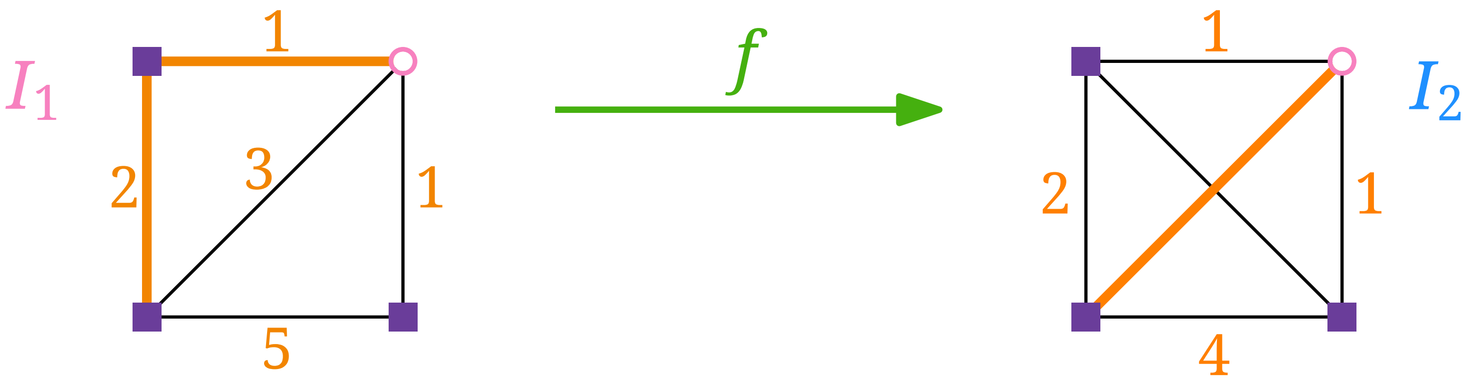
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

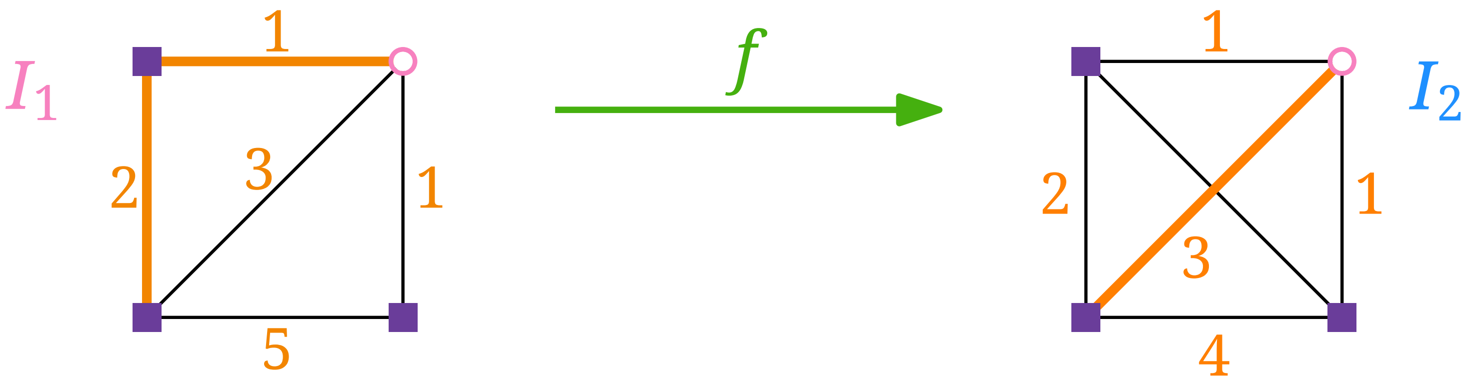
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

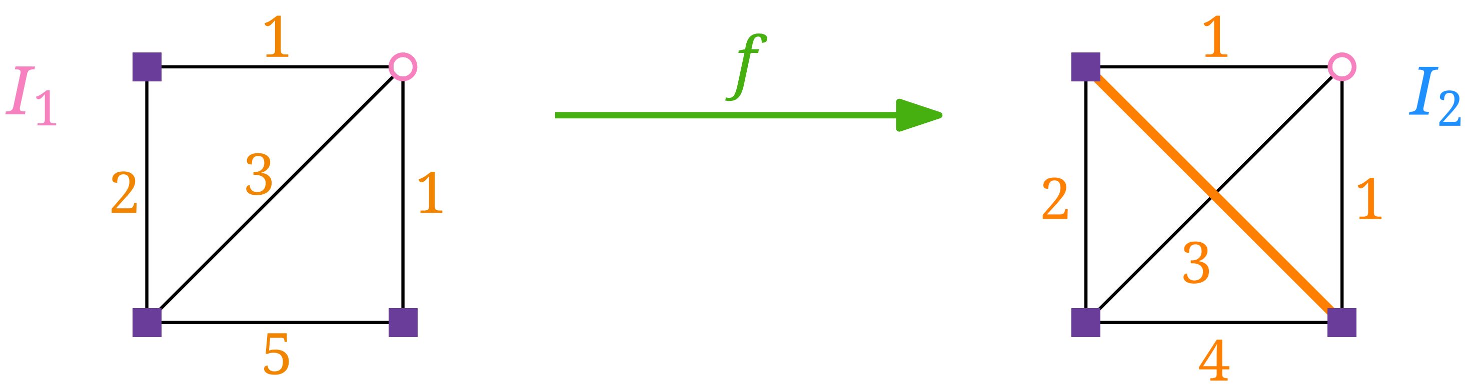
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

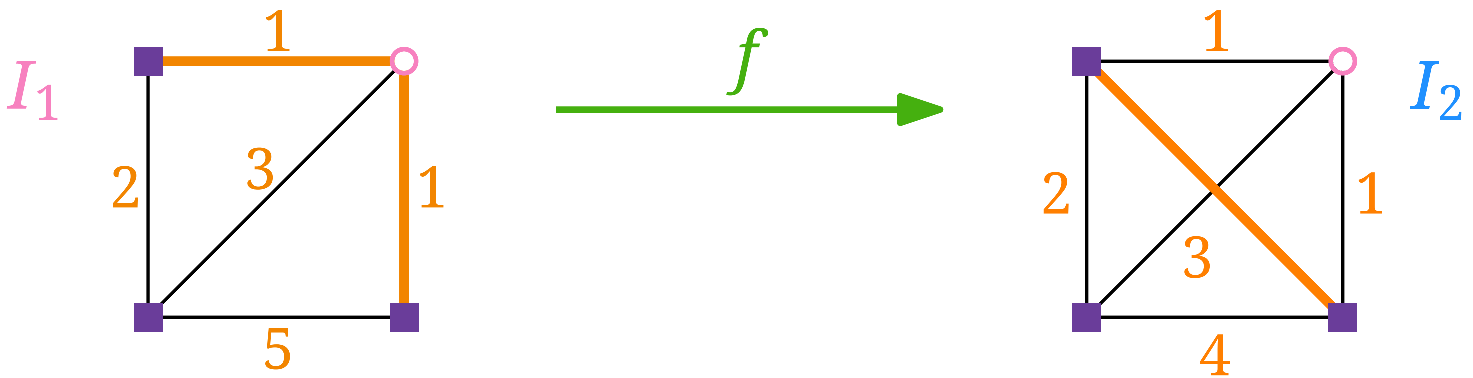
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:

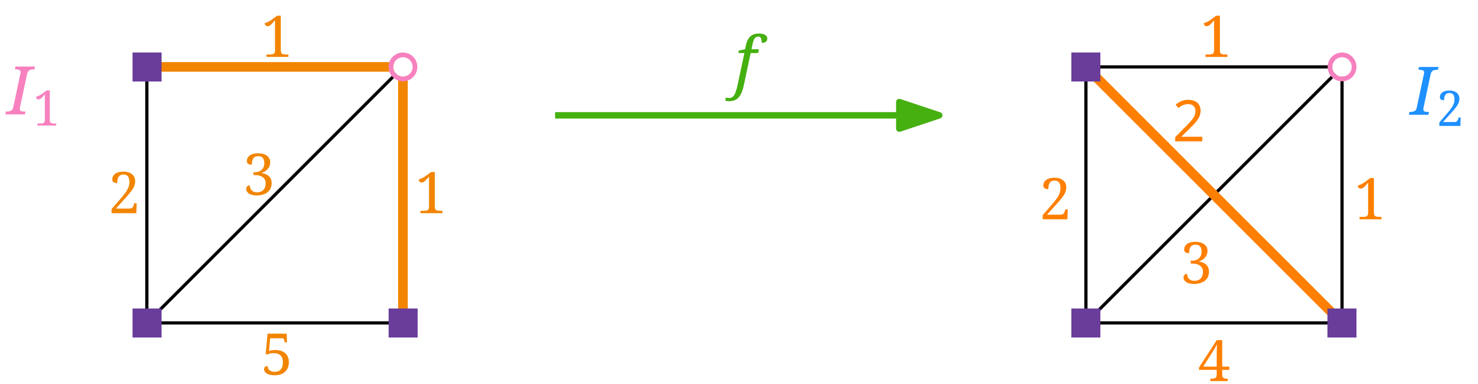
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



MetricSteinerTree

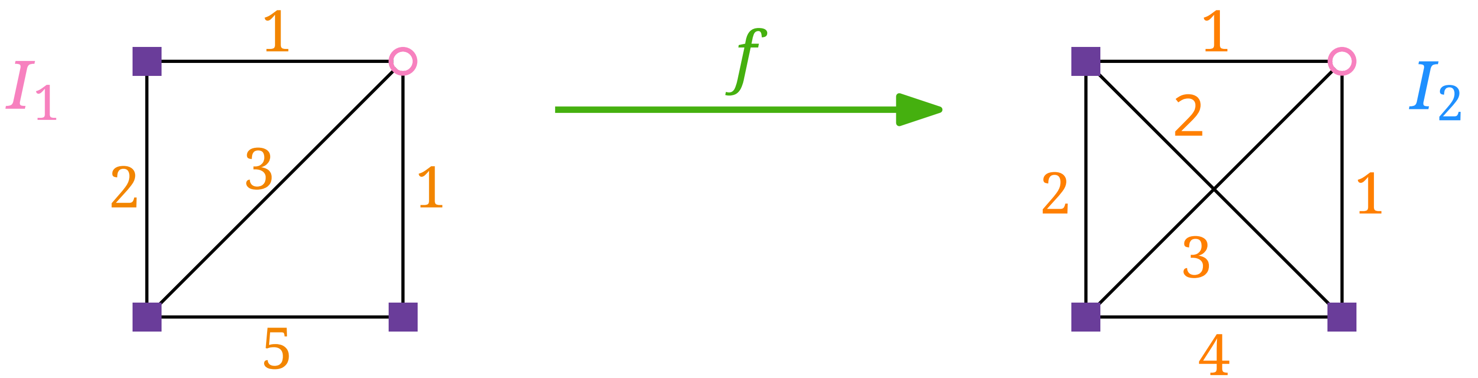
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of SteinerTree:
Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \,\dot\cup\, S$

Metric instance $I_2 := f(I_1)$:
Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1
$c_2(u, v)$:= Length of a shortest u–v path in G_1.

G_2 is the "metric closure" of G_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $\quad I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1, partition $V = T \mathbin{\dot\cup} S$

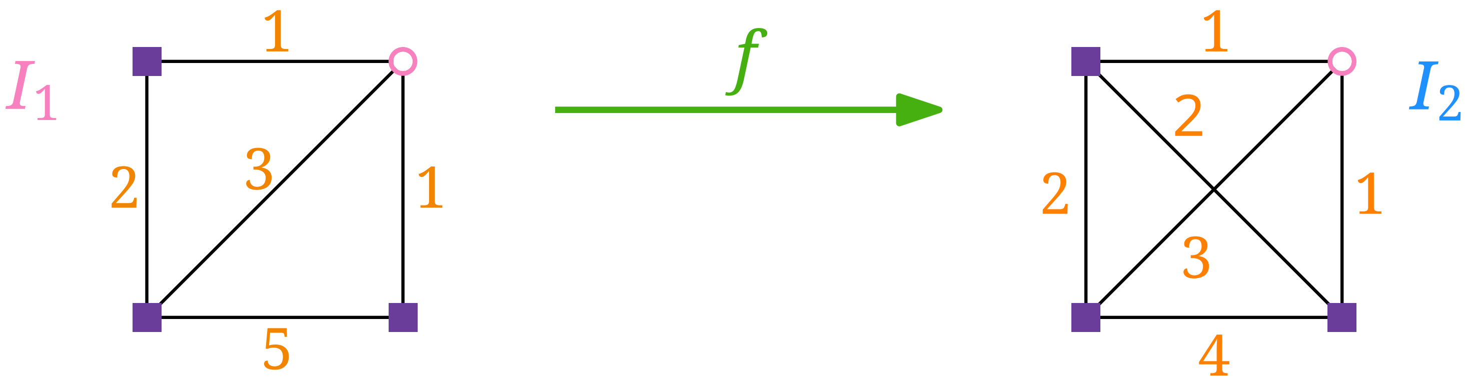
Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ Length of a shortest u–v path in G_1.

$c_2(u, v) \leq c_1(u, v)$ for every edge $(u, v) \in E_1$.

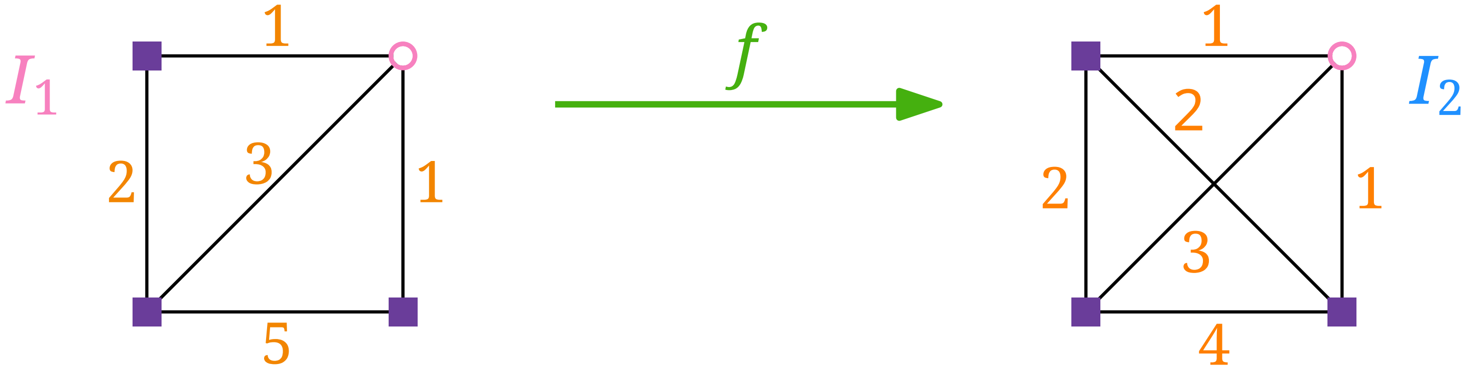
G_2 is the "metric closure" of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

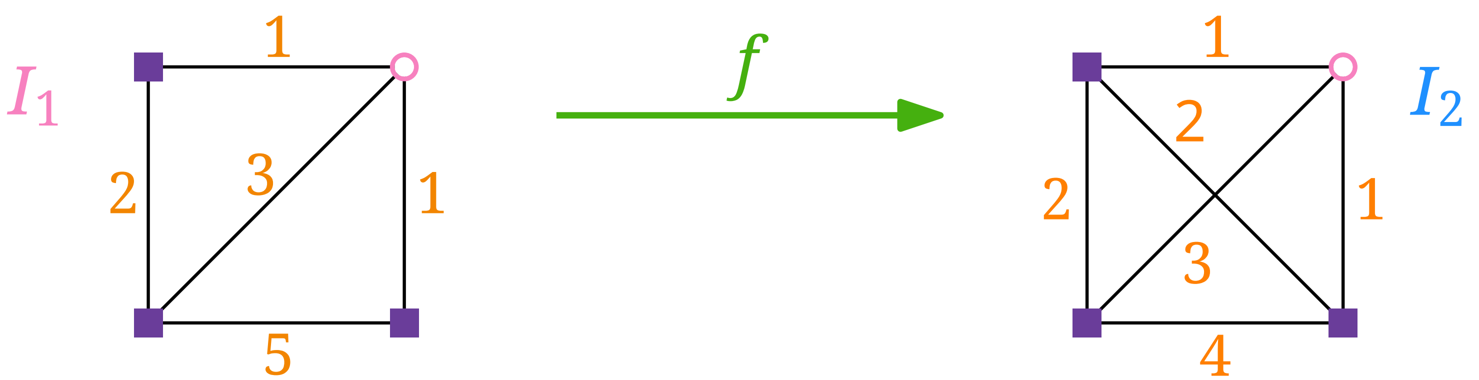


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

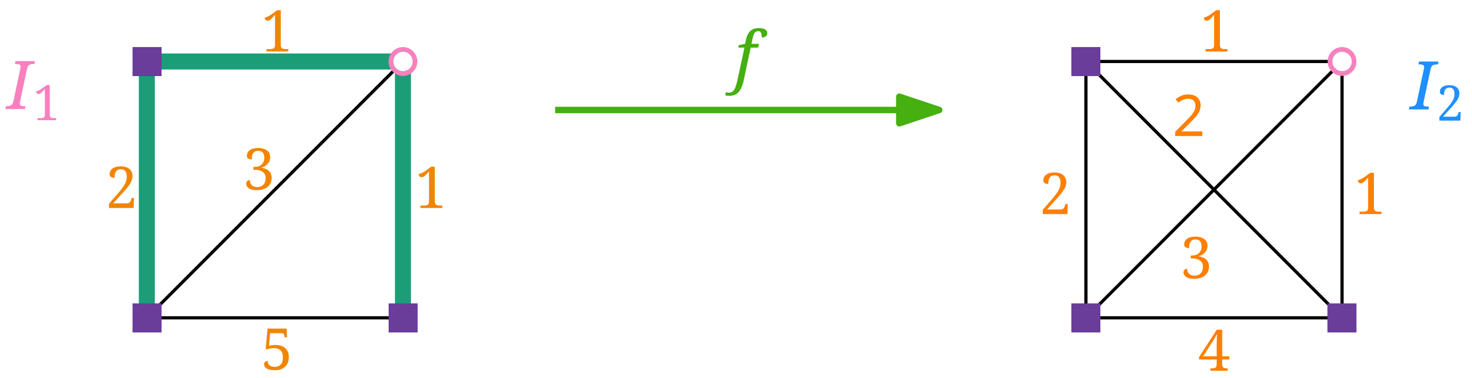


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.



MetricSteinerTree

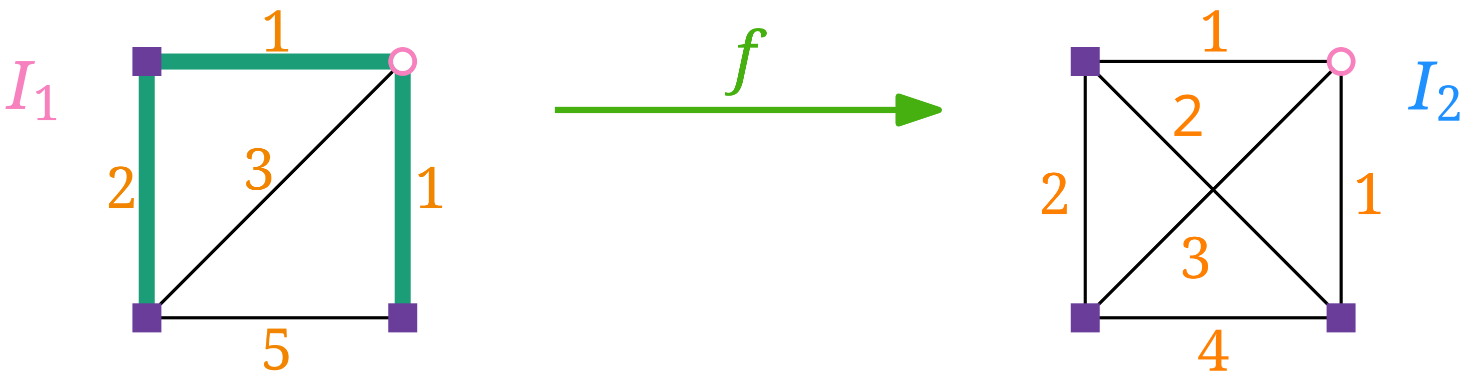
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.



MetricSteinerTree

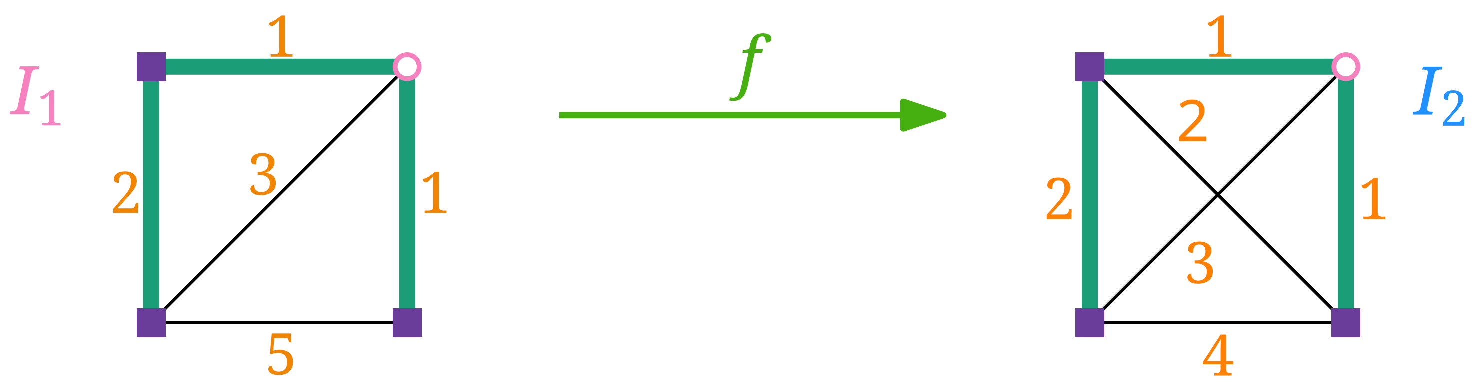
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

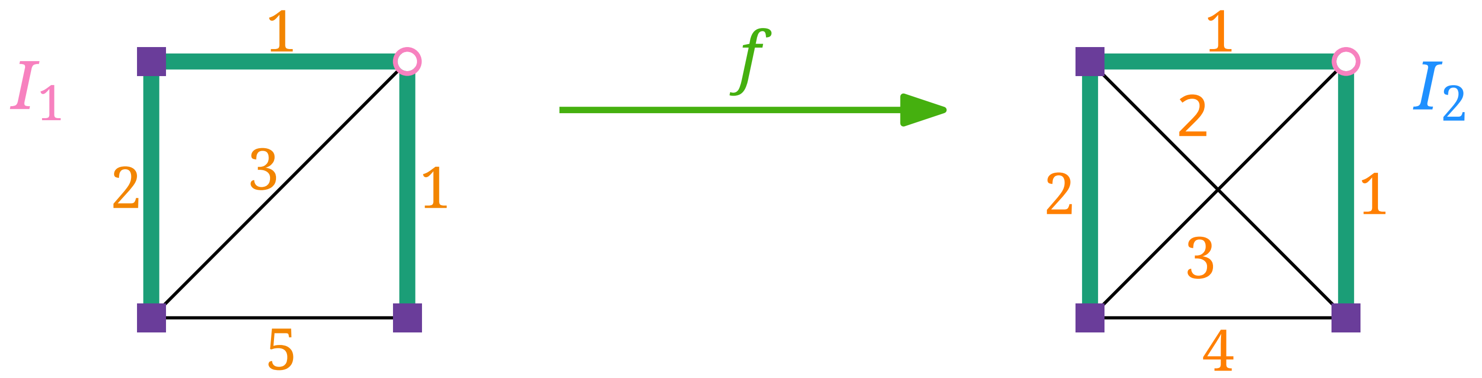
Proof. $\mathrm{OPT}(I_2) \le \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.

$\mathrm{OPT}(I_2)$



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

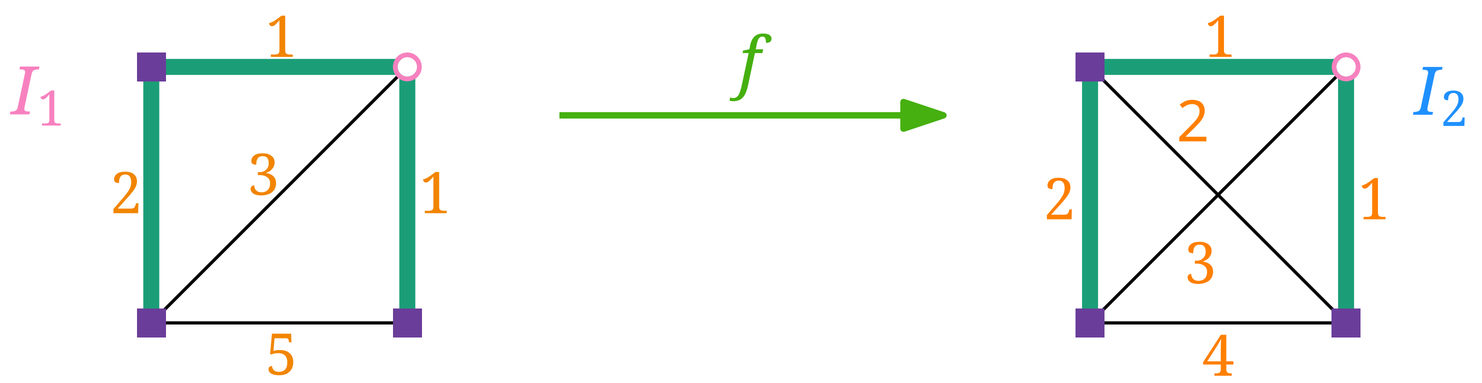
Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.

$\mathrm{OPT}(I_2) \leq c_2(B^*)$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

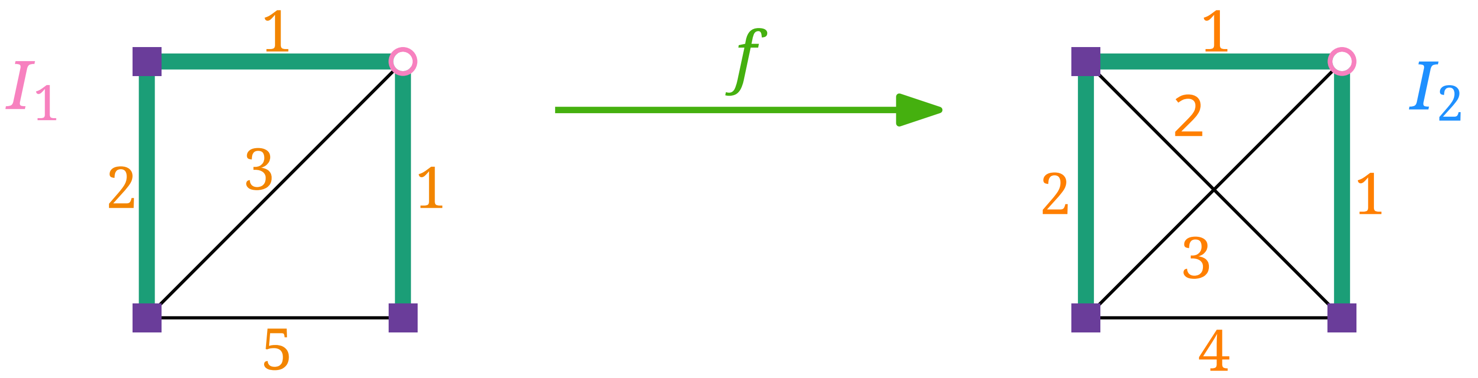
Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.

$\mathrm{OPT}(I_2) \leq c_2(B^*) \leq c_1(B^*)$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

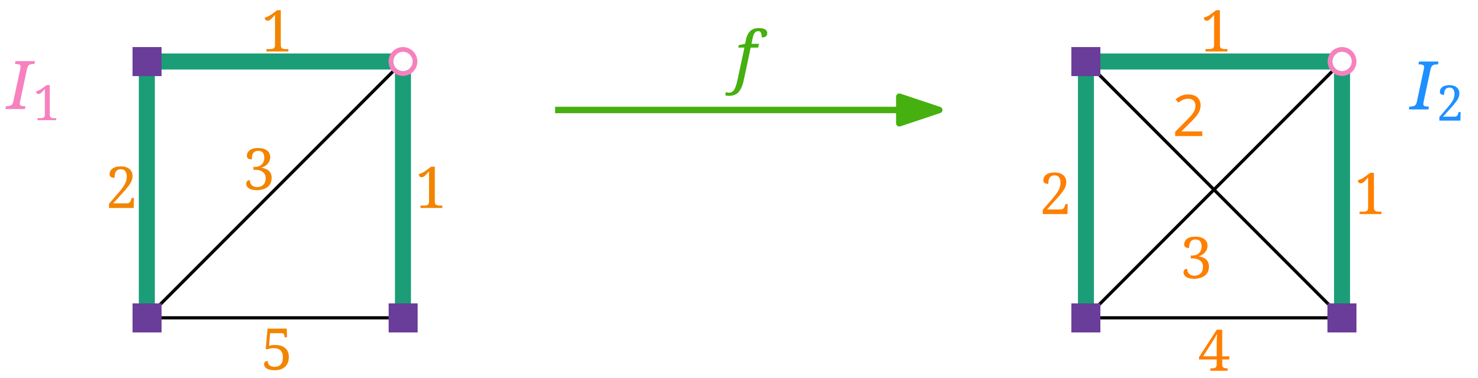
Proof. $\mathrm{OPT}(I_2) \leq \mathrm{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1.

Note that B^* is also a feasible solution for I_2:

$E_1 \subseteq E_2$ and the vertex sets V, T, S are the same.

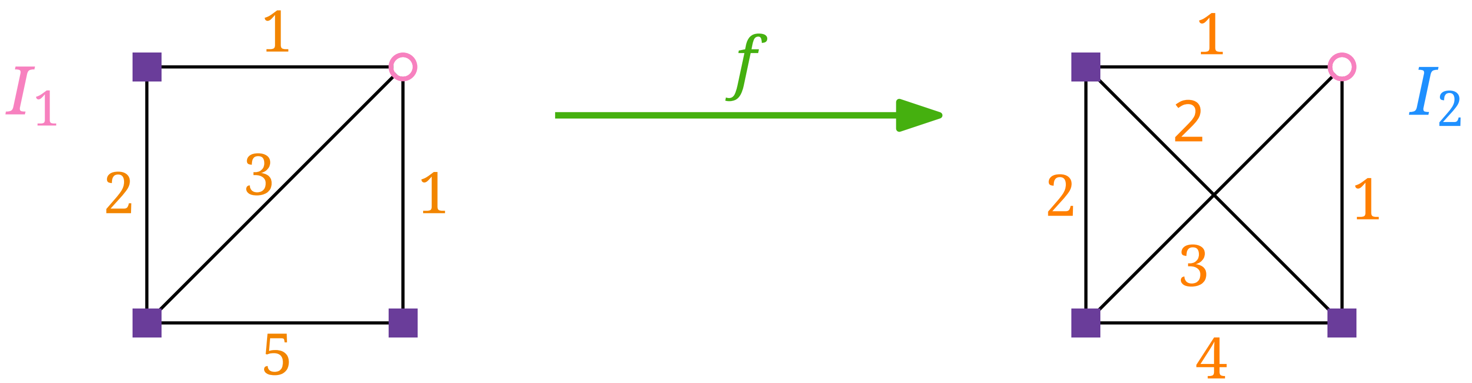
$\mathrm{OPT}(I_2) \leq c_2(B^*) \leq c_1(B^*) = \mathrm{OPT}(I_1)$



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

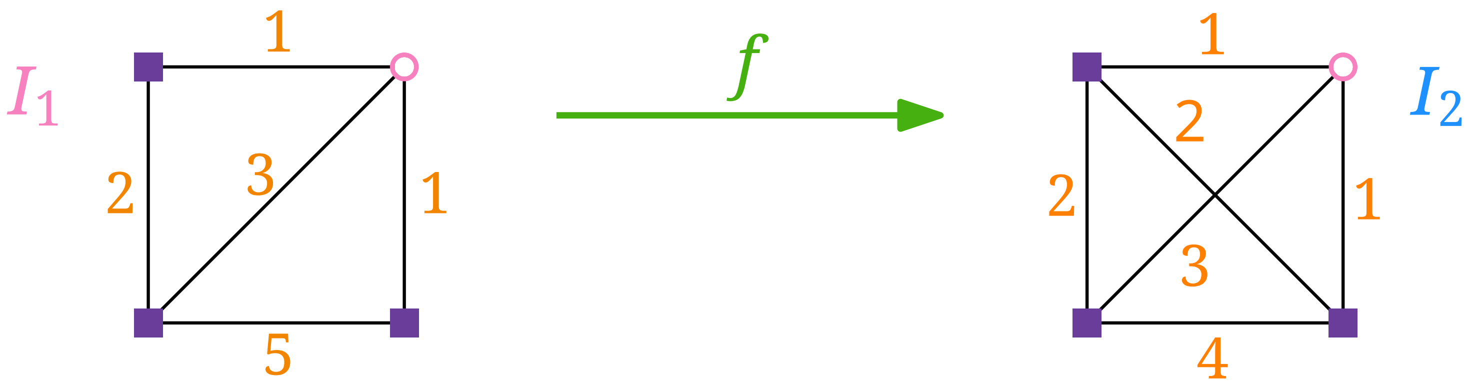


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

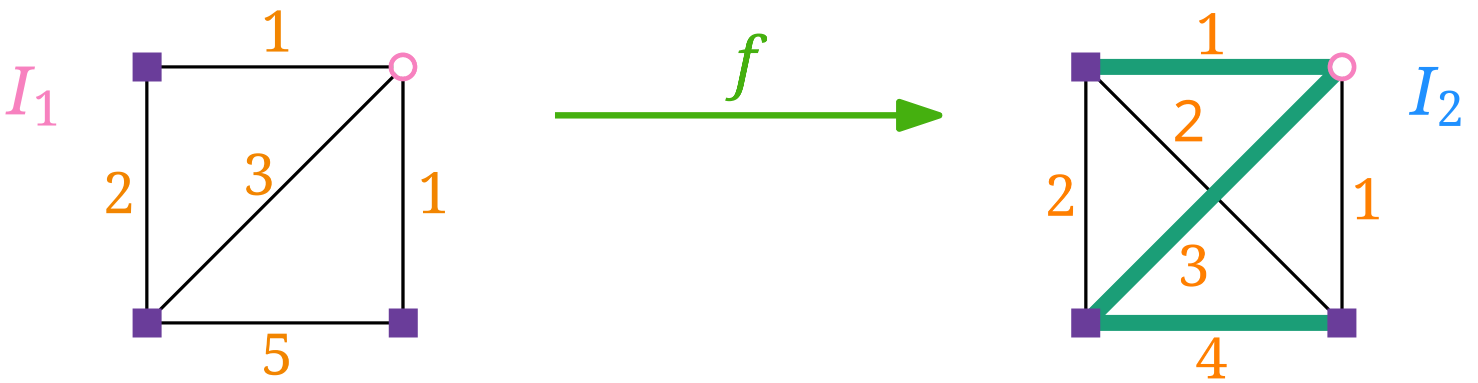


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

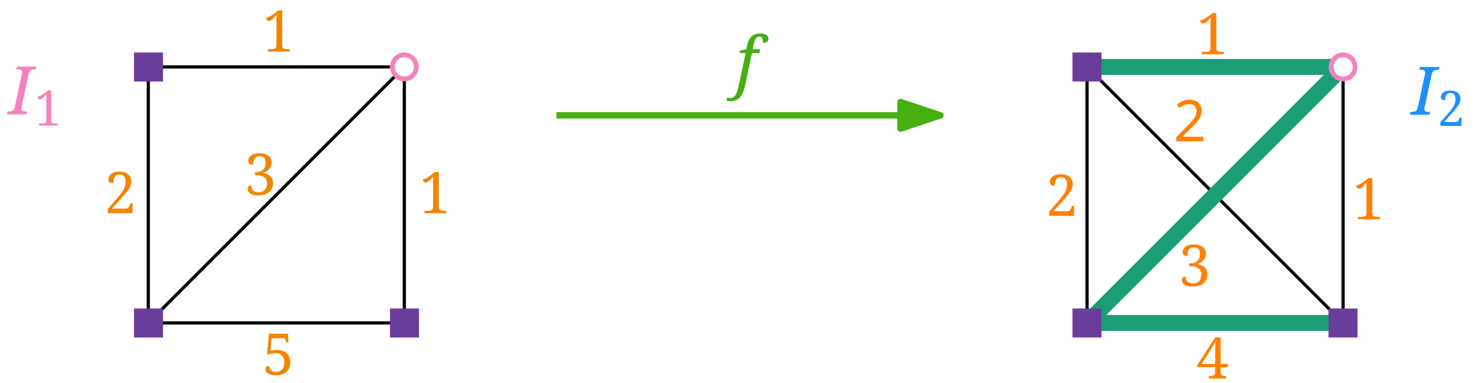


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1.



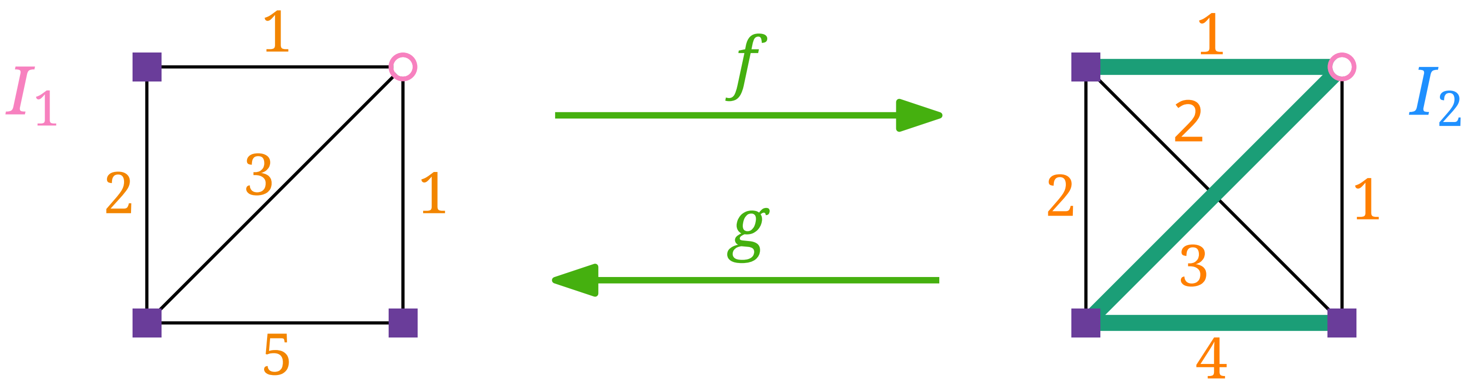
MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1.

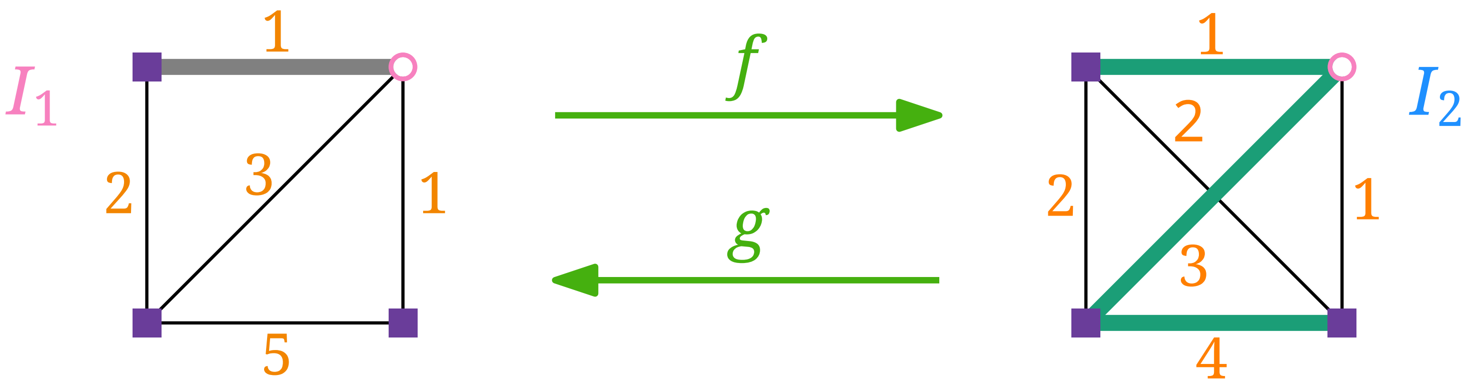


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1.

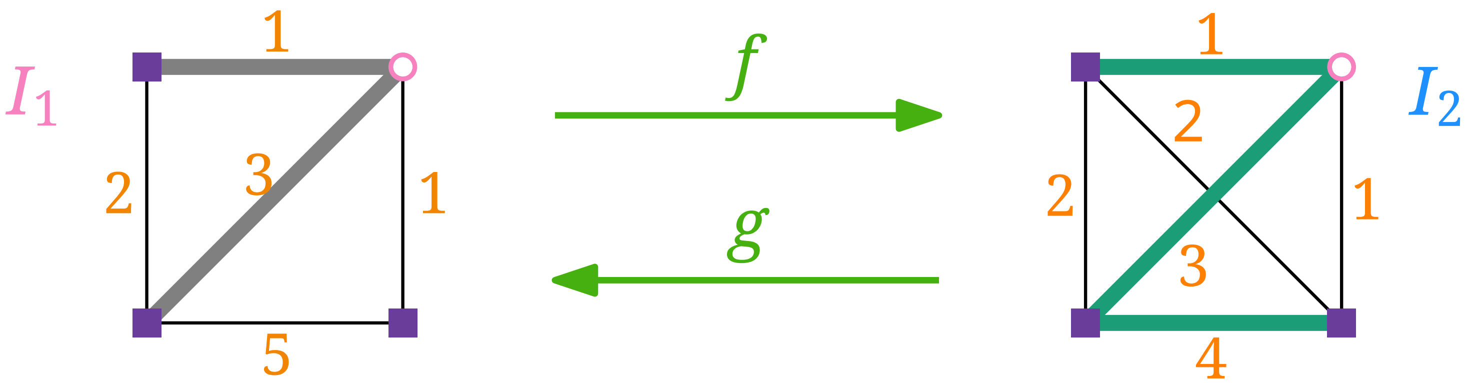


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1.

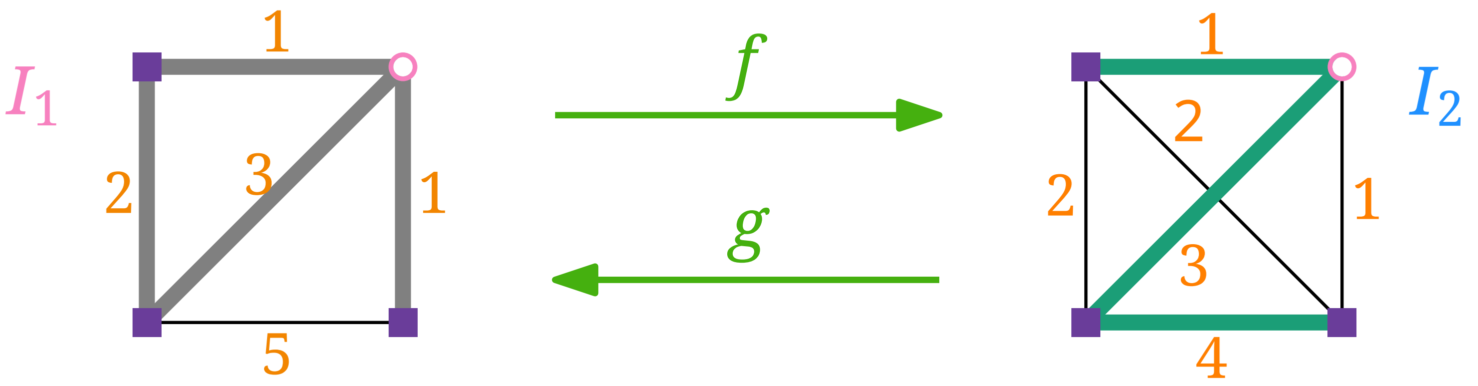


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1.

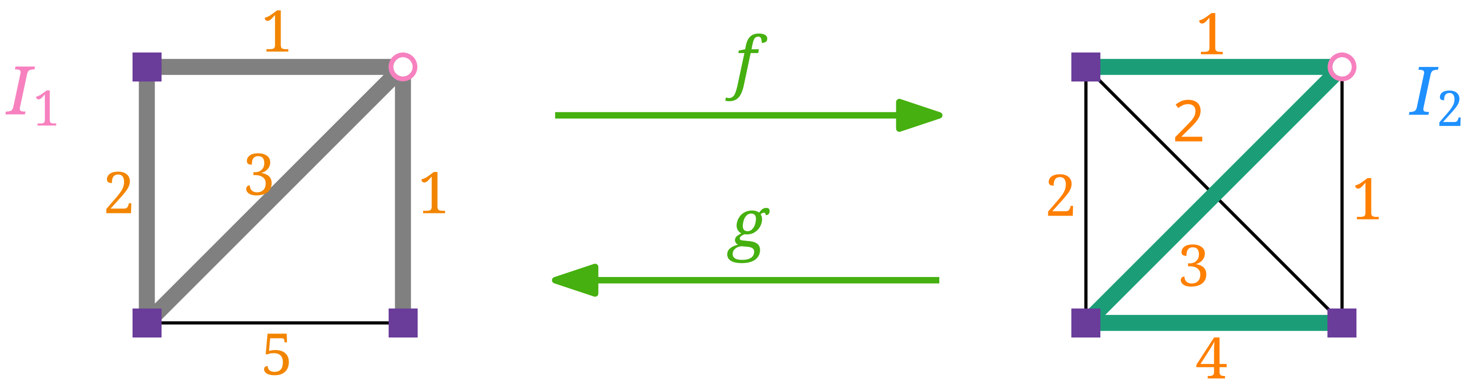


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.



MetricSteinerTree

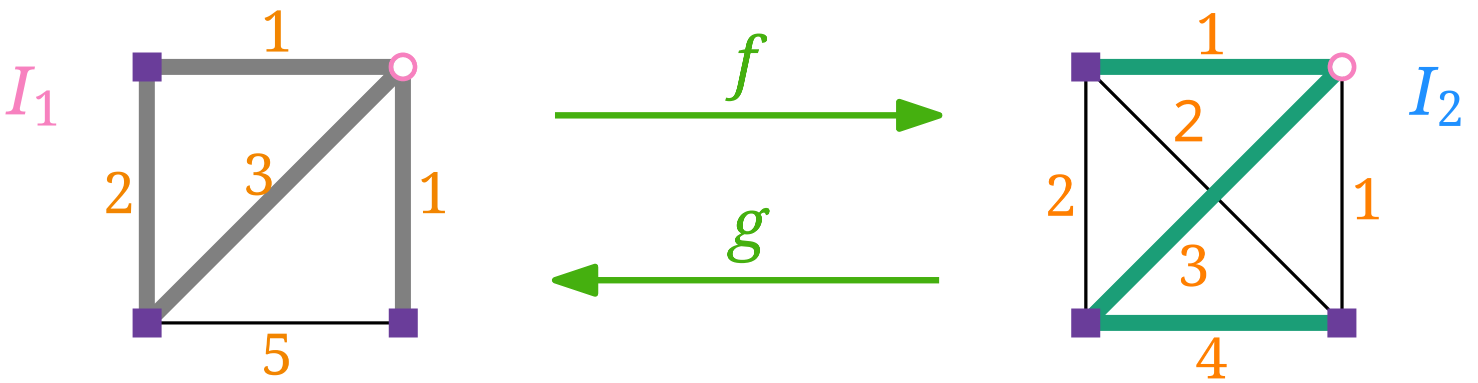
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$

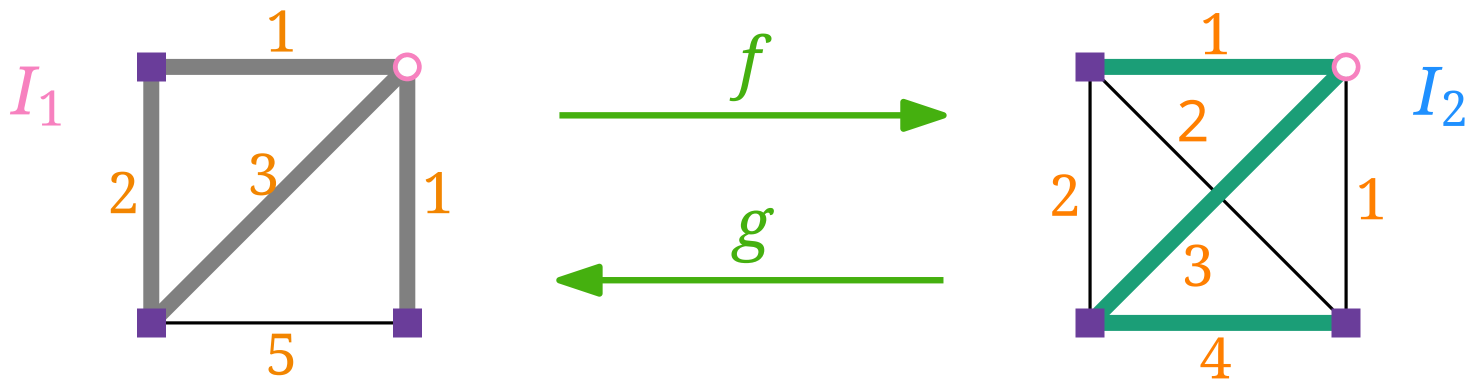


MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.
Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.
$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals



MetricSteinerTree

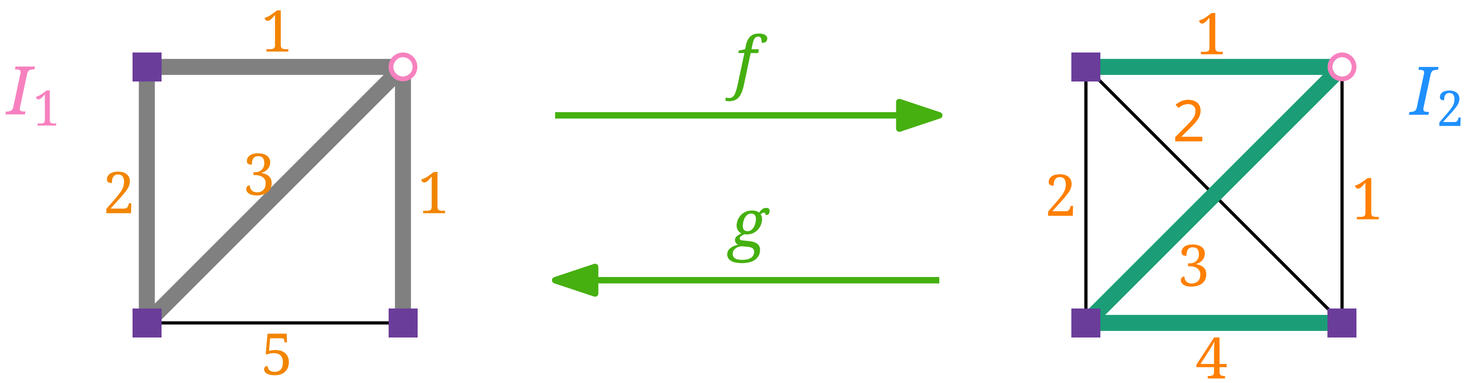
Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals ; maybe not a tree.



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

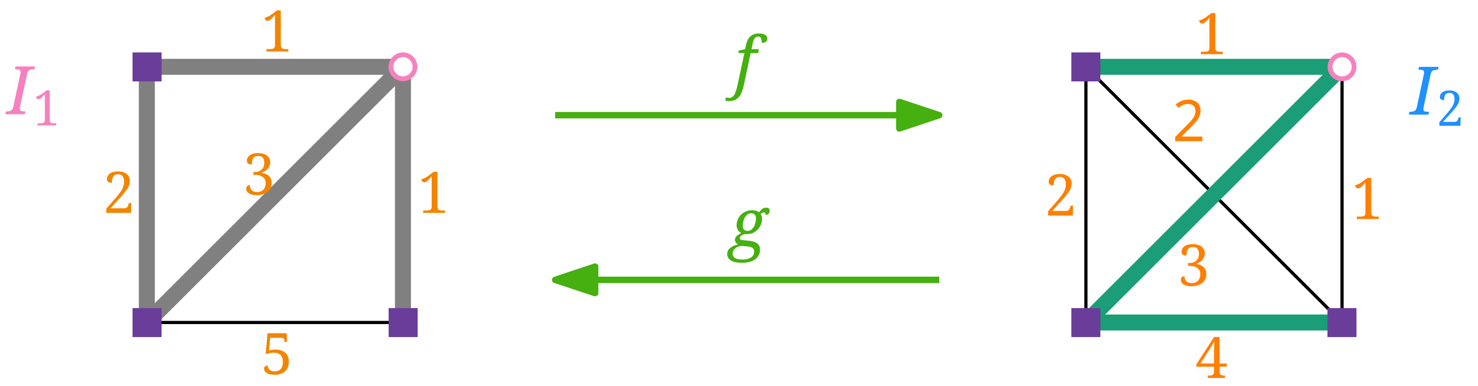
Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals ; maybe not a tree.

Consider spanning tree B_1 of G_1'



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

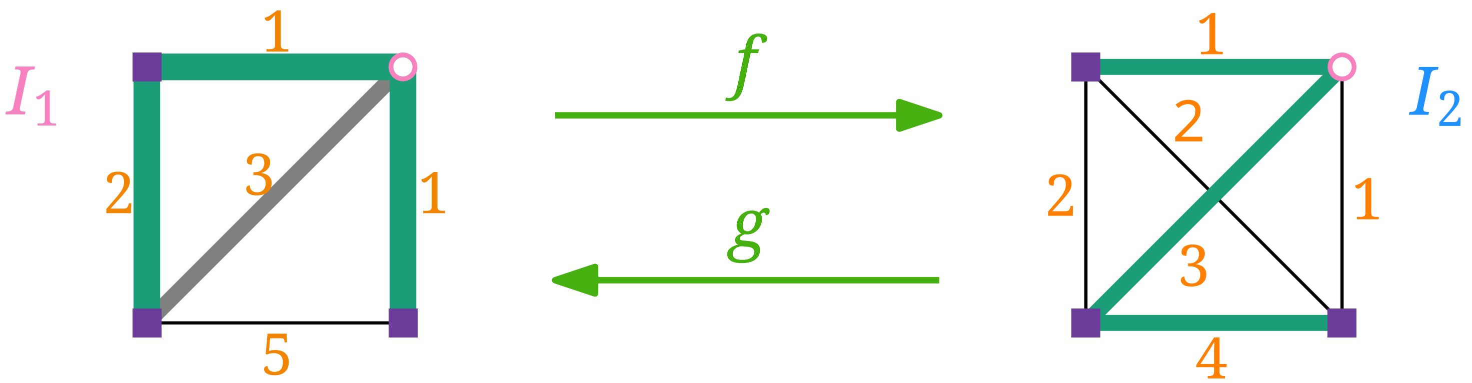
Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals ; maybe not a tree.

Consider spanning tree B_1 of G_1'



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

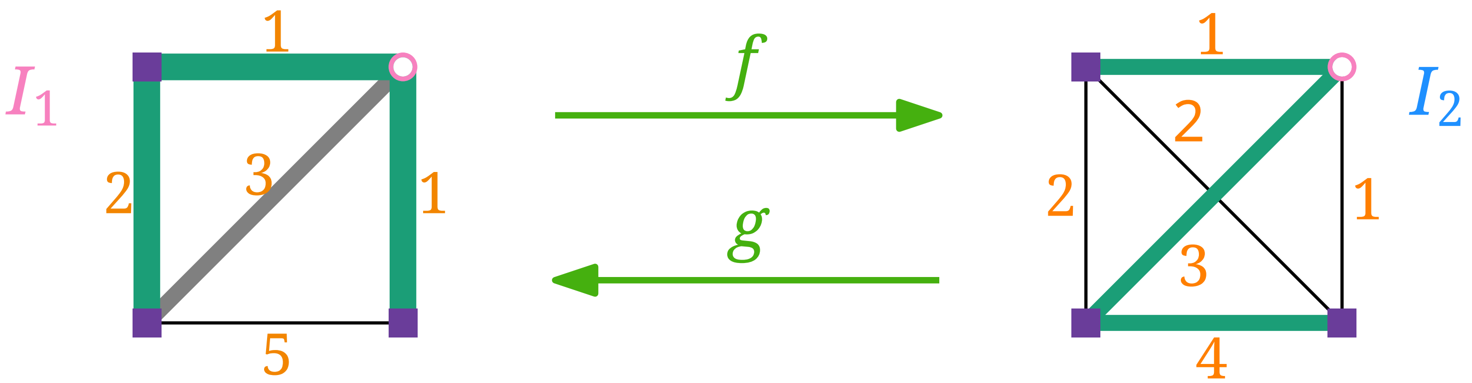
Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals ; maybe not a tree.

Consider spanning tree B_1 of G_1' $\rightsquigarrow$ Steiner tree B_1 of G_1



MetricSteinerTree

Theorem. There is an approximation-preserving reduction from SteinerTree to MetricSteinerTree.

Proof. (2) Mapping g $\quad s \xleftarrow{g} t$

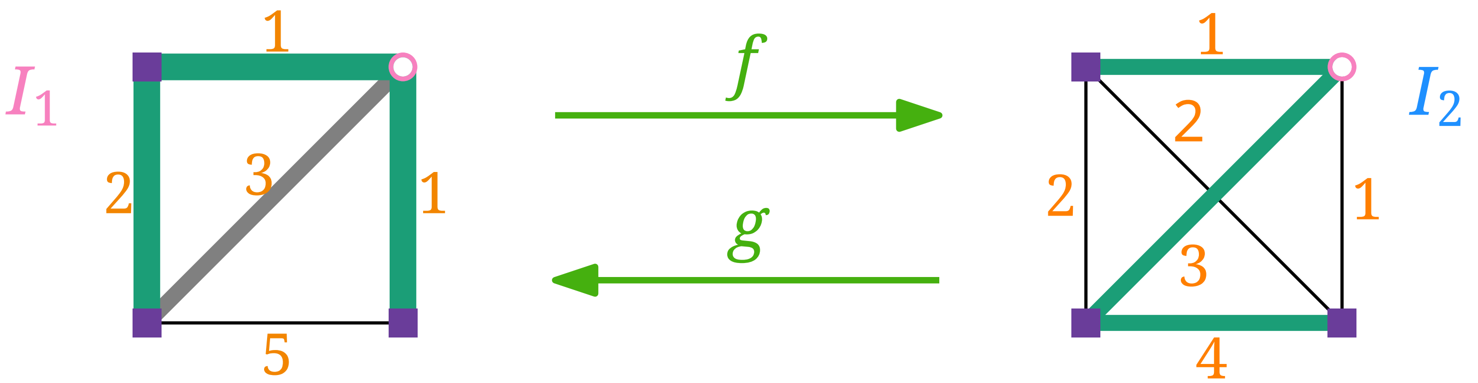
Let B_2 be a Steiner tree of G_2.

Construct $G_1' \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest u–v path in G_1. Keep ≤ 1 copy per edge.

$c_1(G_1') \leq c_2(B_2)$; G_1' connects all terminals ; maybe not a tree.

Consider spanning tree B_1 of G_1' $\rightsquigarrow$ Steiner tree B_1 of G_1

Thus $c_1(B_1) \leq c_1(G_1') \leq c_2(B_2)$, i.e. $\mathrm{obj}_{\Pi_1}(I_1, s) \leq \mathrm{obj}_{\Pi_2}(I_2, t)$



2-Approximation for SteinerTree

2-Approximation for MetricSteinerTree

Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

2-Approximation for MetricSteinerTree

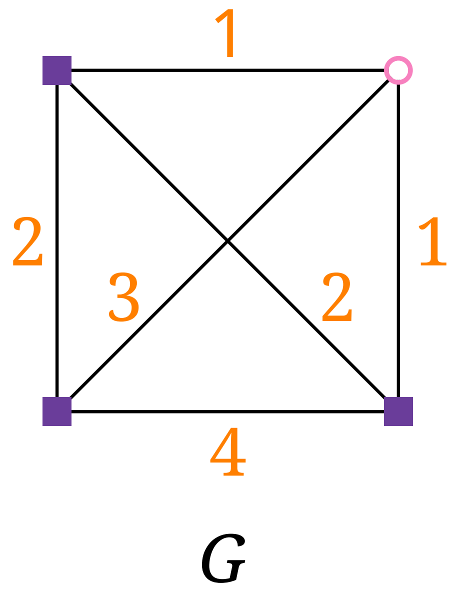
Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

Theorem. The algorithm is a 2-approximation for MetricSteinerTree.

2-Approximation for MetricSteinerTree

Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

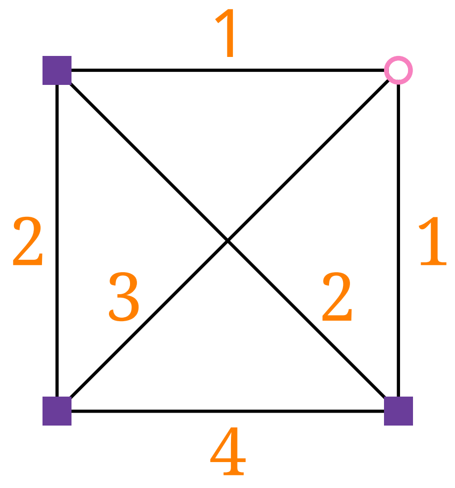
Theorem. The algorithm is a 2-approximation for MetricSteinerTree.



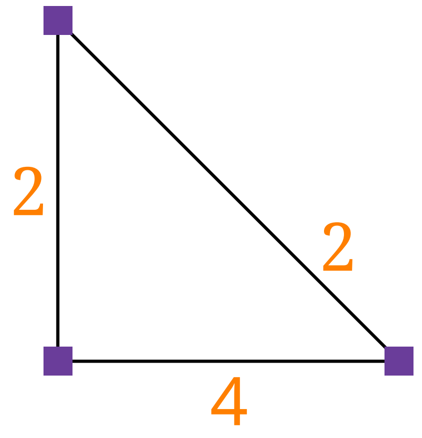
2-Approximation for MetricSteinerTree

Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

Theorem. The algorithm is a 2-approximation for MetricSteinerTree.



G

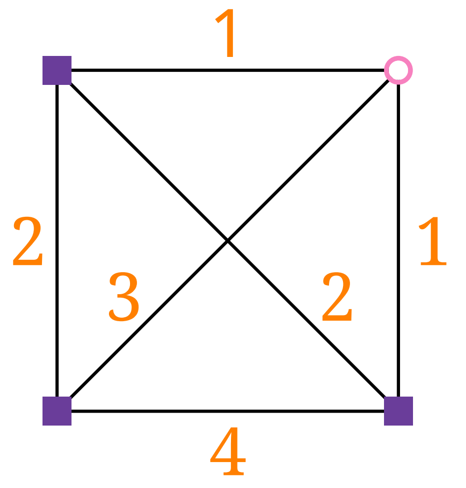


$G[T]$

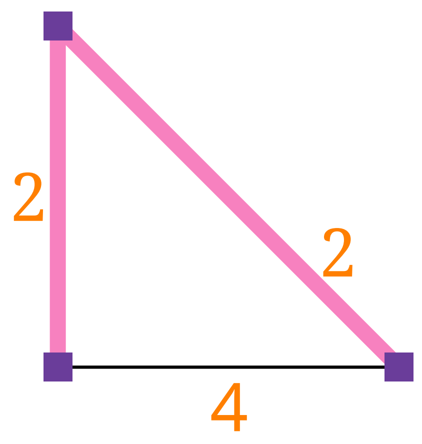
2-Approximation for MetricSteinerTree

Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

Theorem. The algorithm is a 2-approximation for MetricSteinerTree.



G

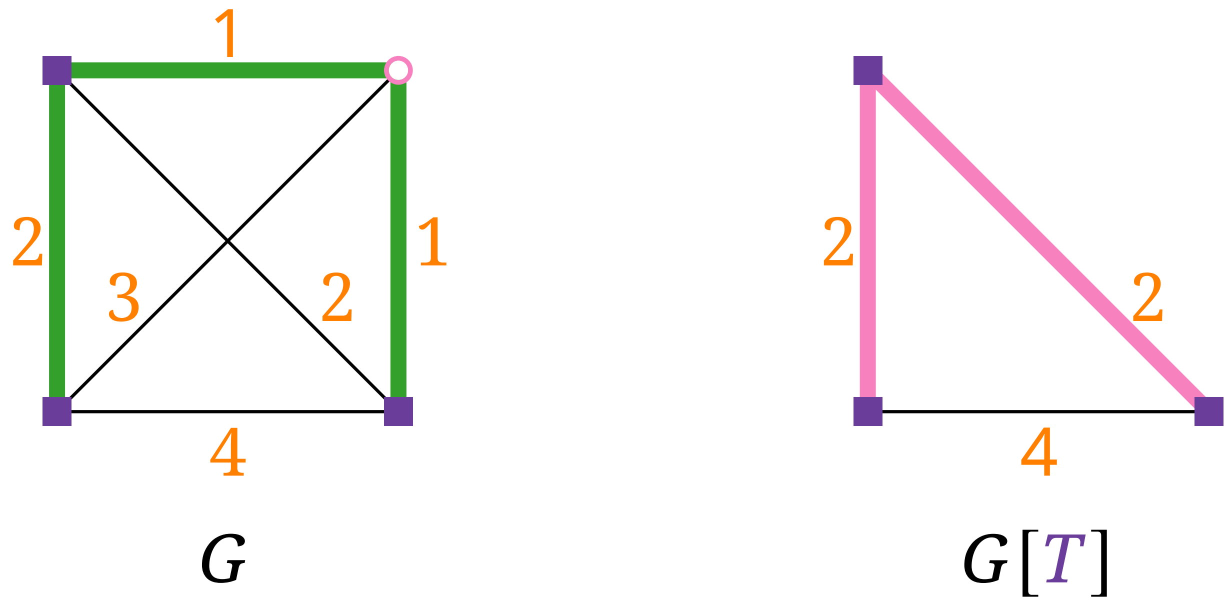


$G[T]$

2-Approximation for MetricSteinerTree

Algorithm: Compute a minimum spanning tree (MST) B in the subgraph $G[T]$ induced by the terminal set T.

Theorem. The algorithm is a 2-approximation for MetricSteinerTree.

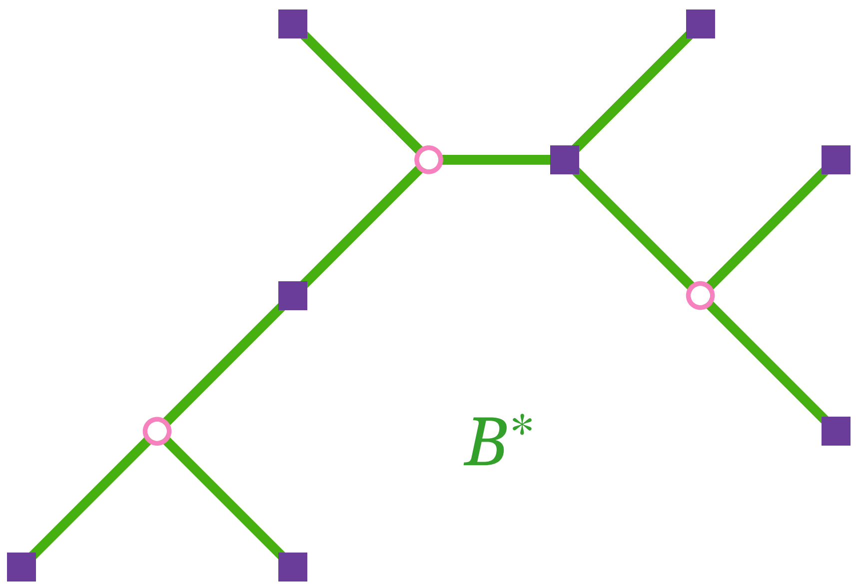


Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Proof of Approximation Factor

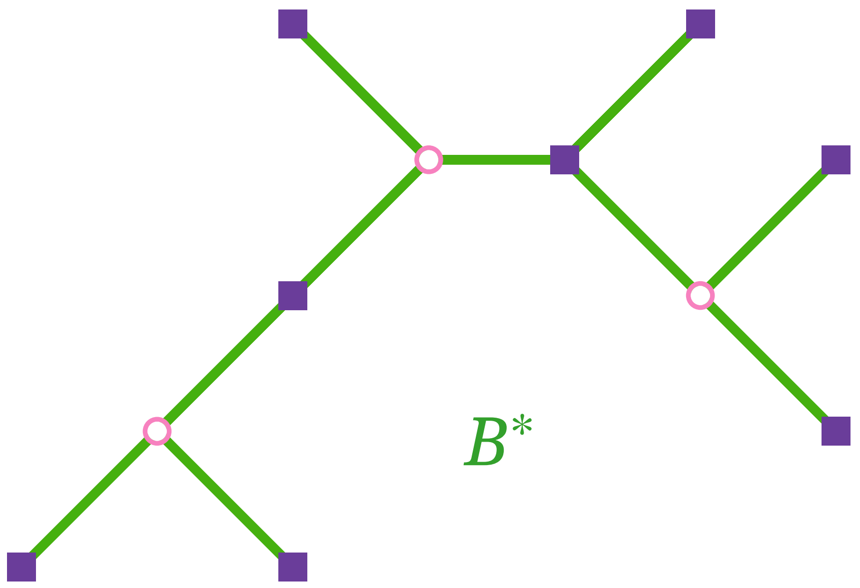
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \text{OPT}$.



Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

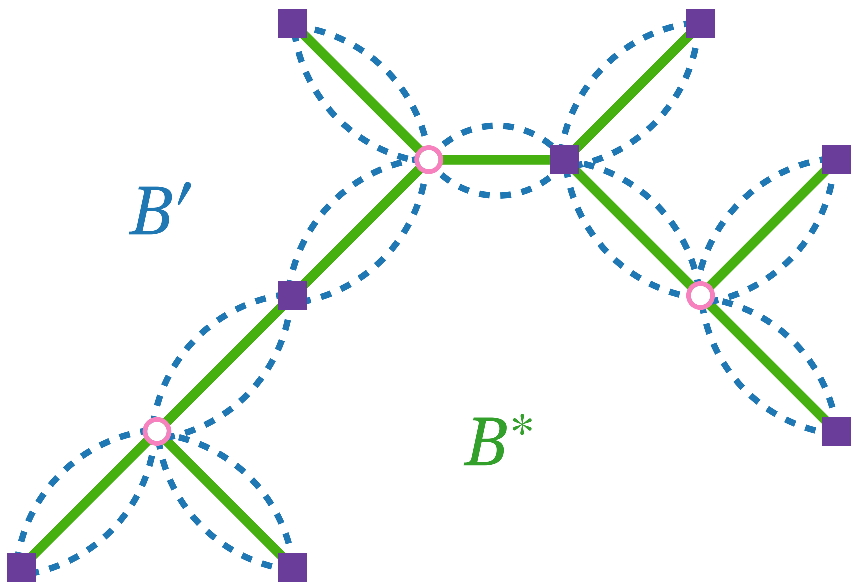
Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.



Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

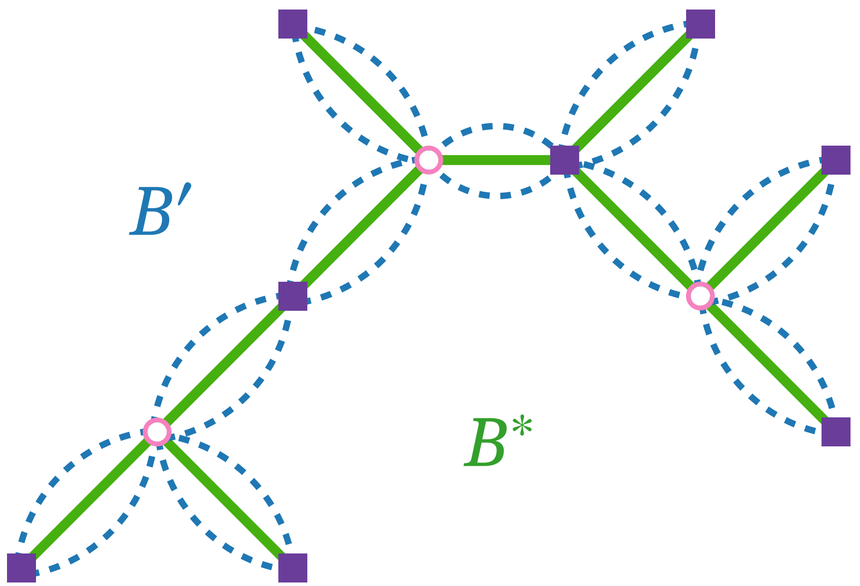


Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B'

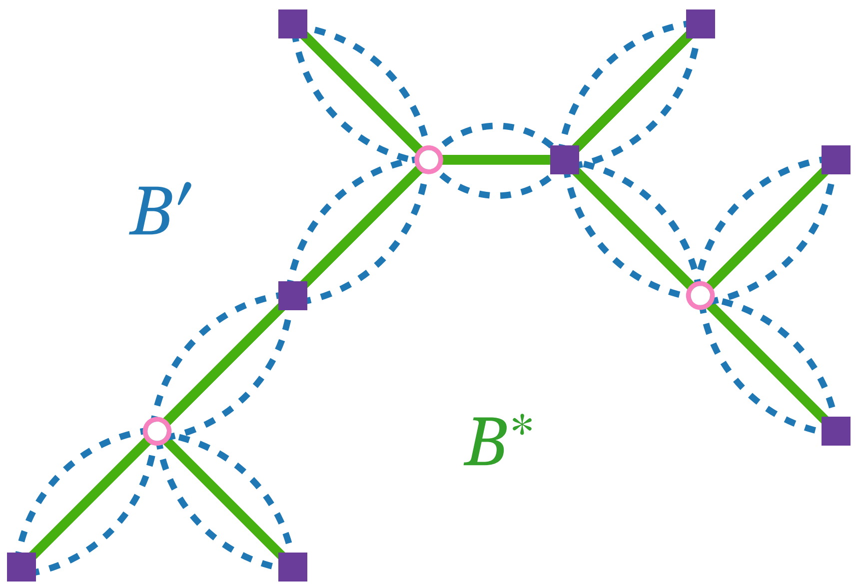


Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

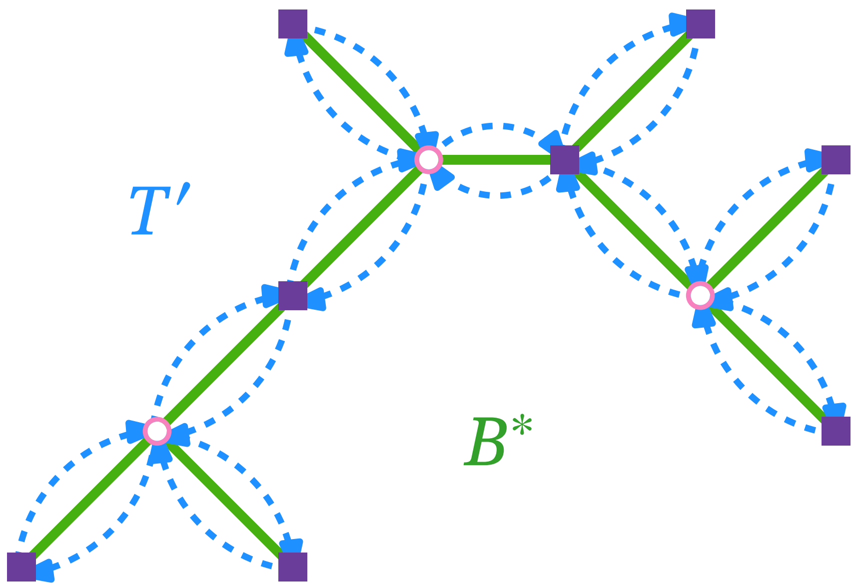


Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$



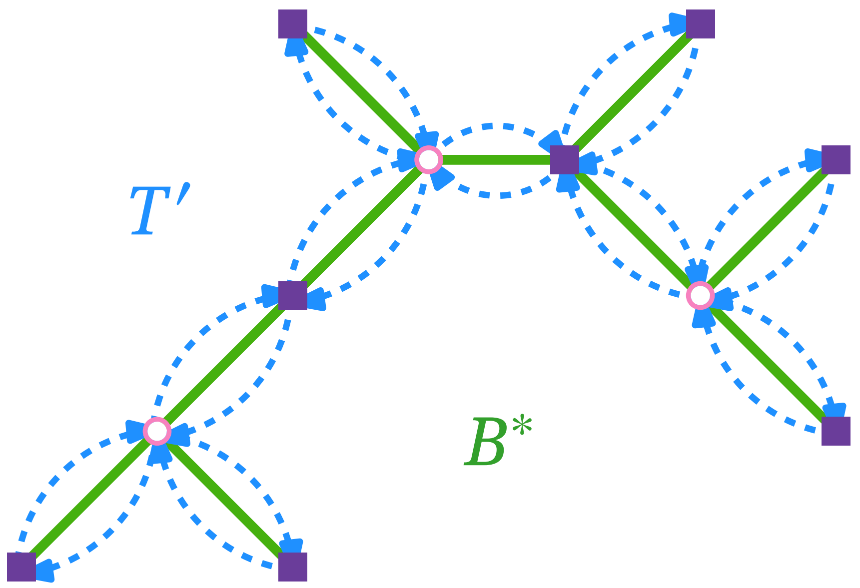
Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.



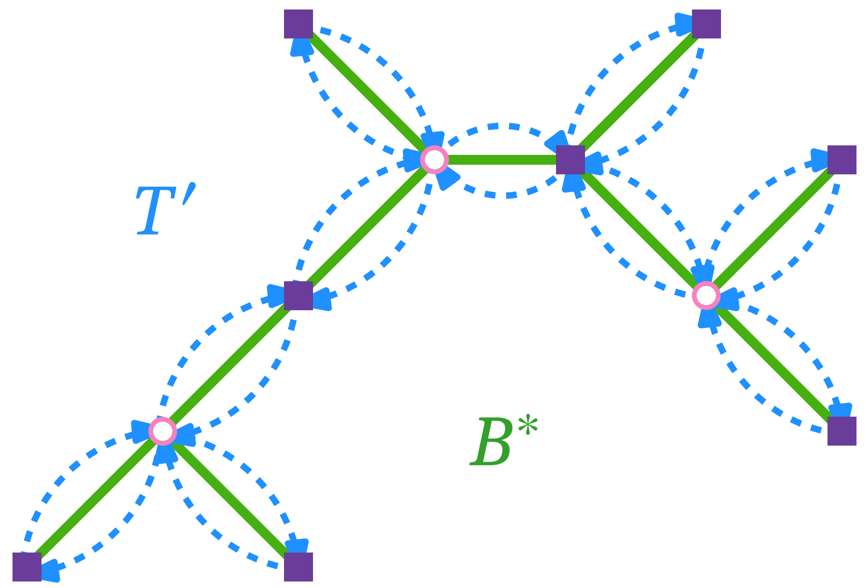
Proof of Approximation Factor

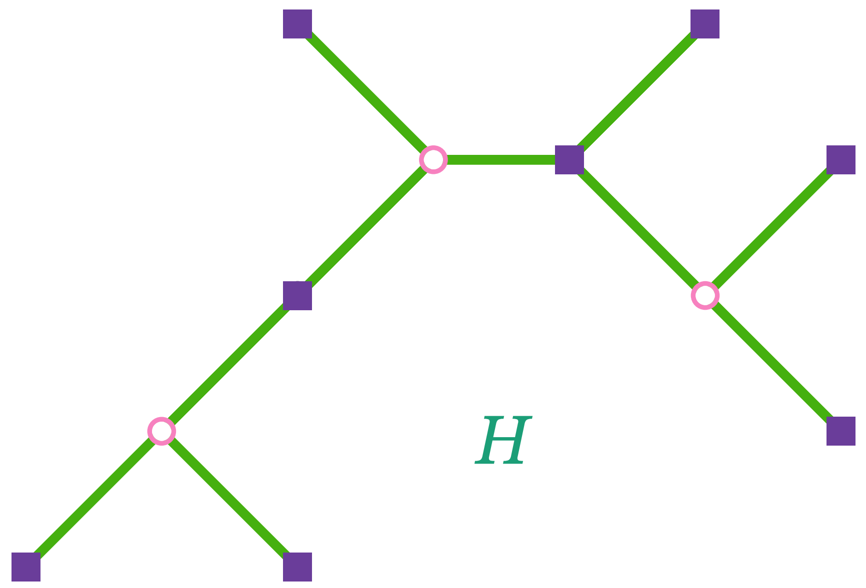
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.





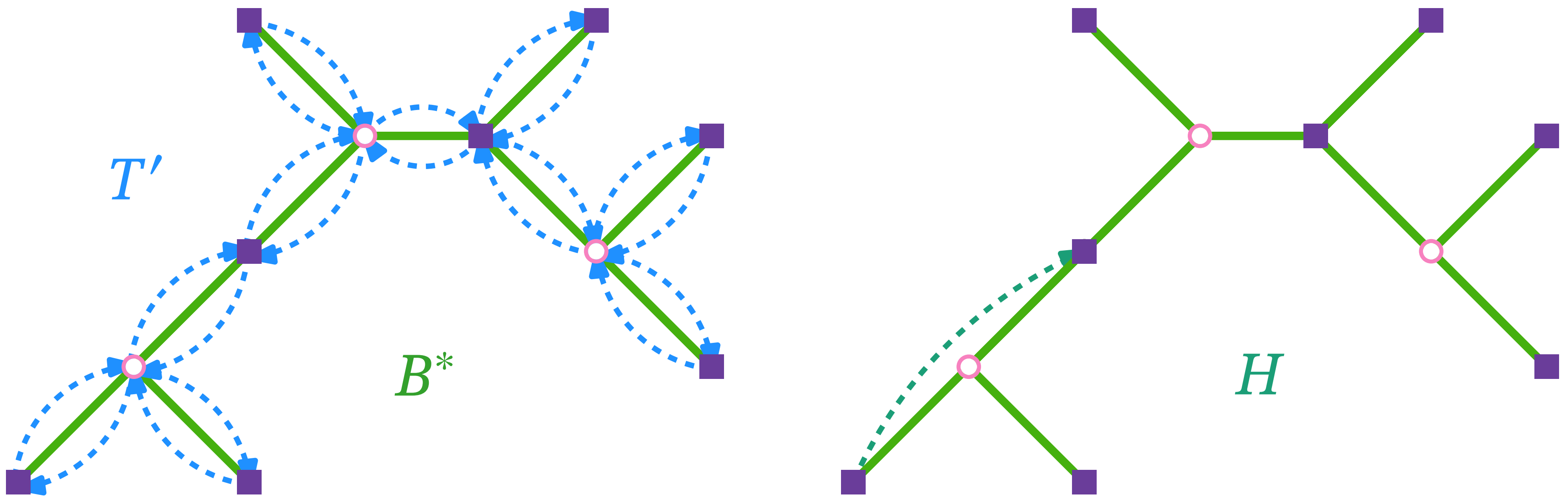
Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad\Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.



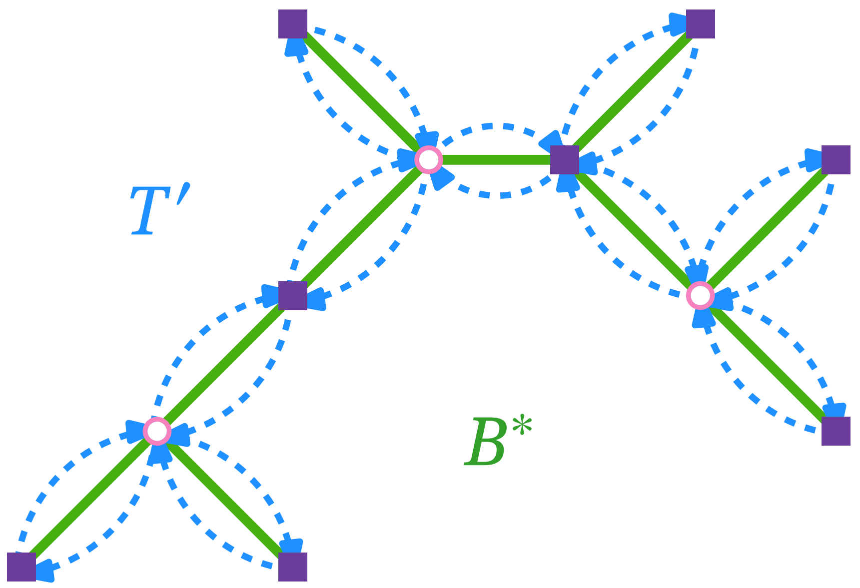
Proof of Approximation Factor

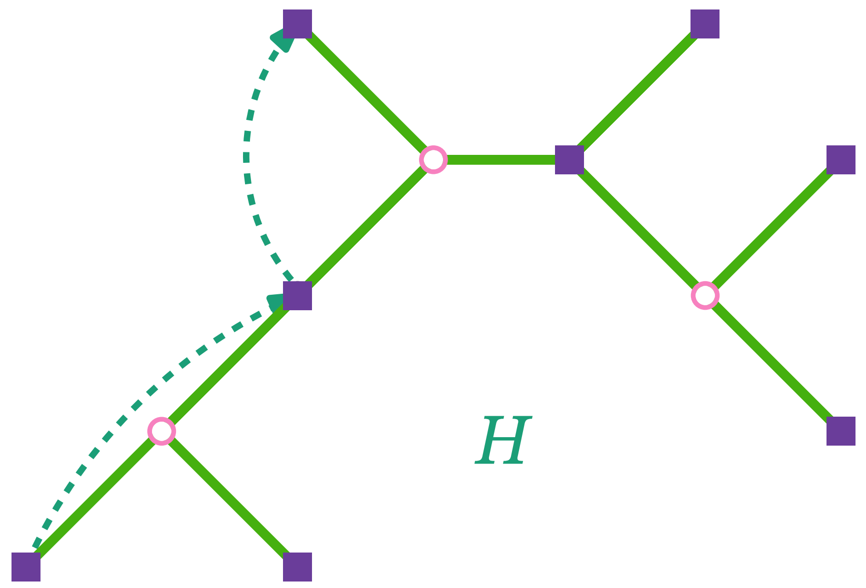
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \text{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in B' $\Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.





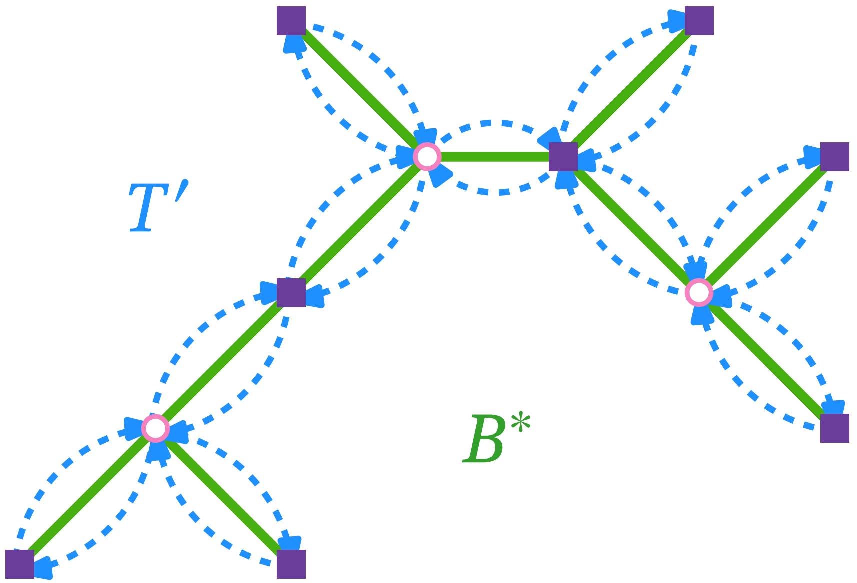
Proof of Approximation Factor

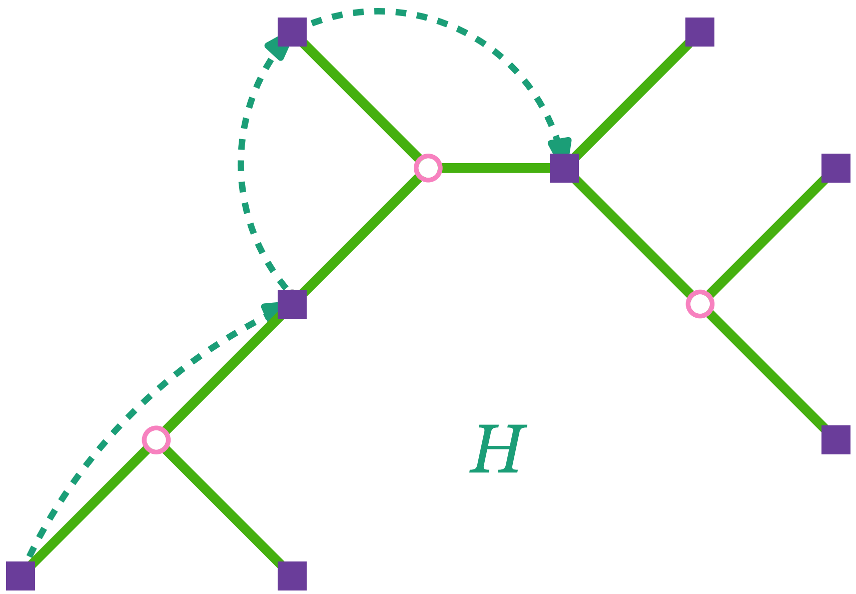
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.





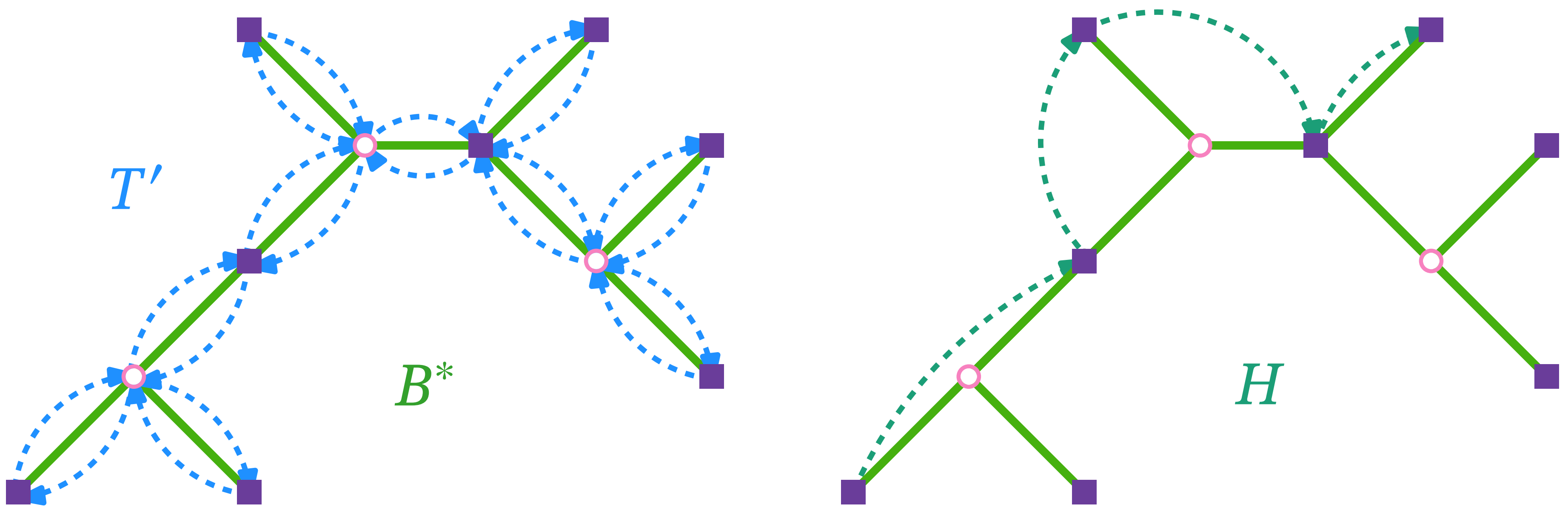
Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.



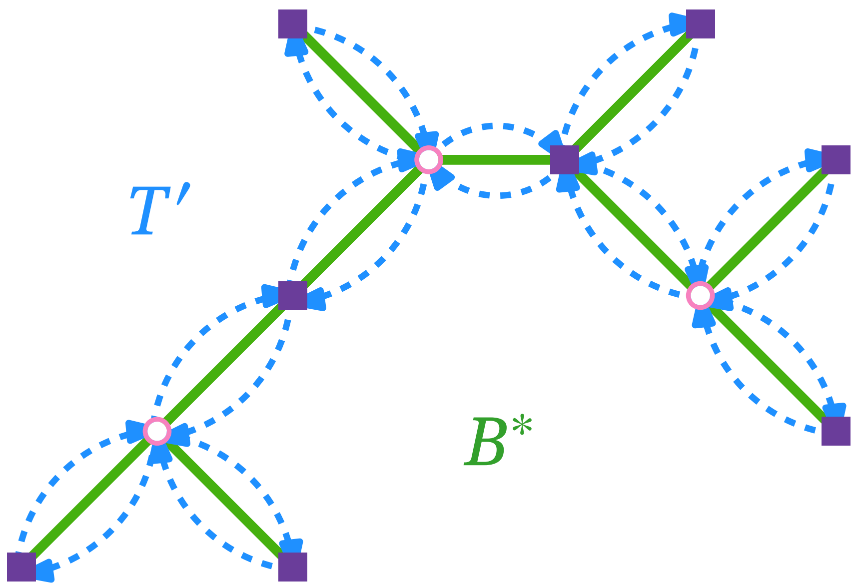
Proof of Approximation Factor

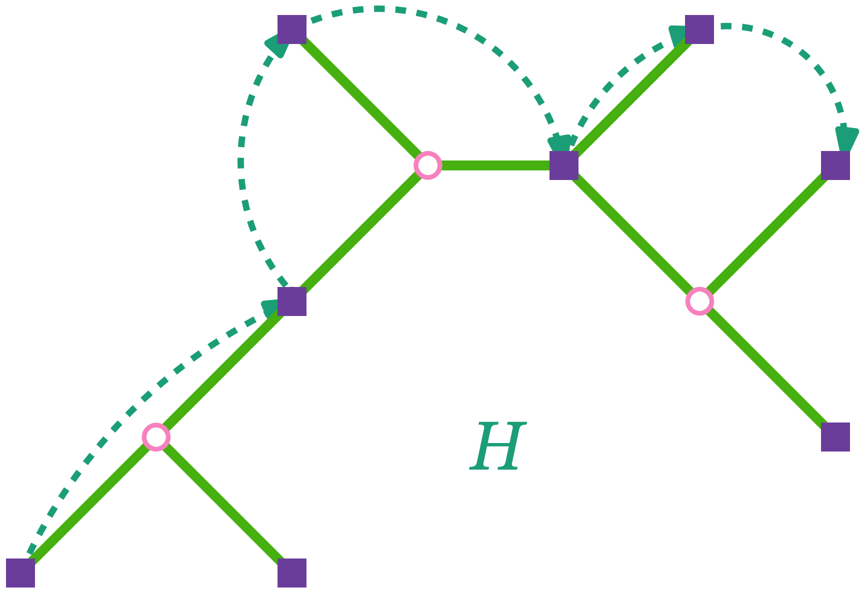
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.





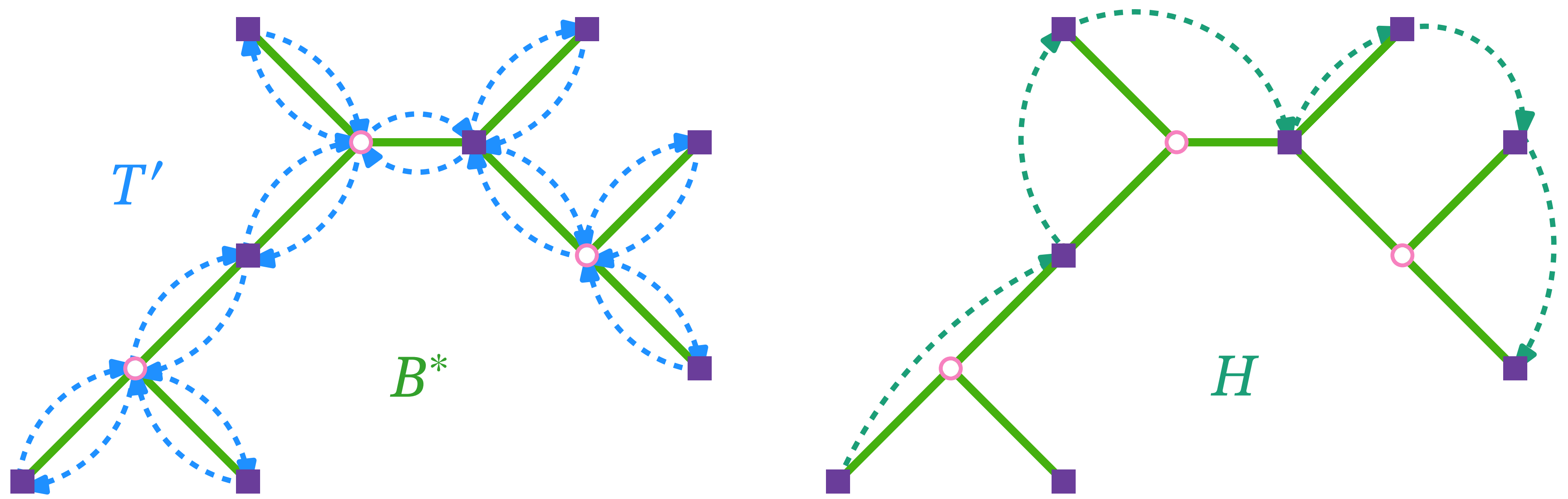
Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.



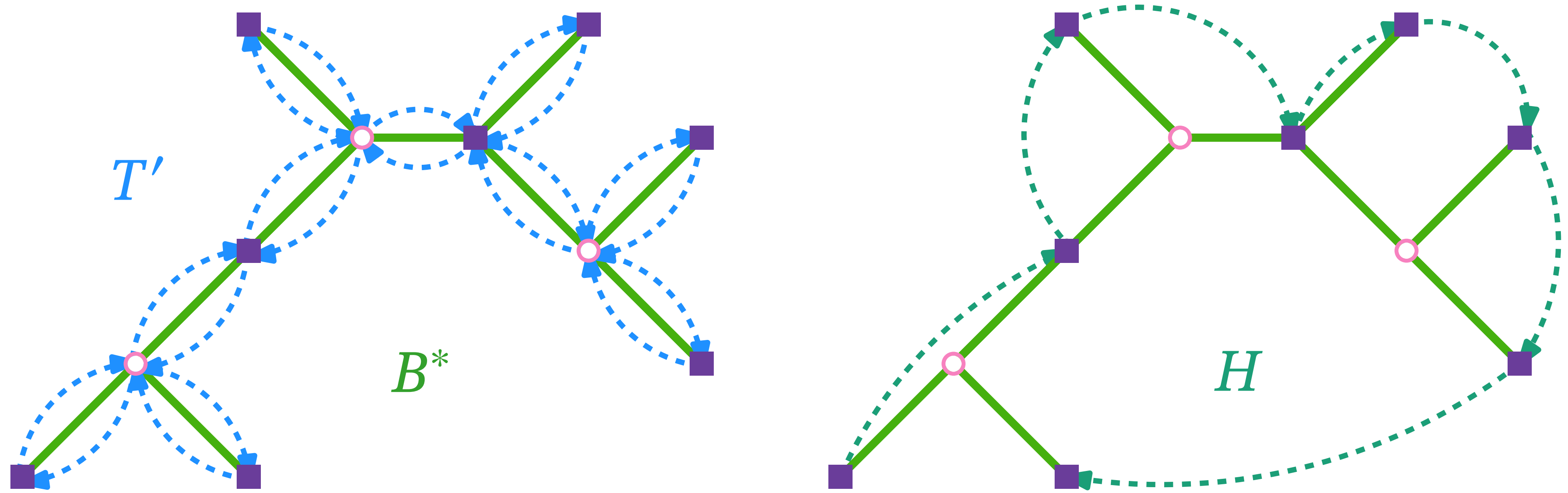
Proof of Approximation Factor

Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and already visited terminals.



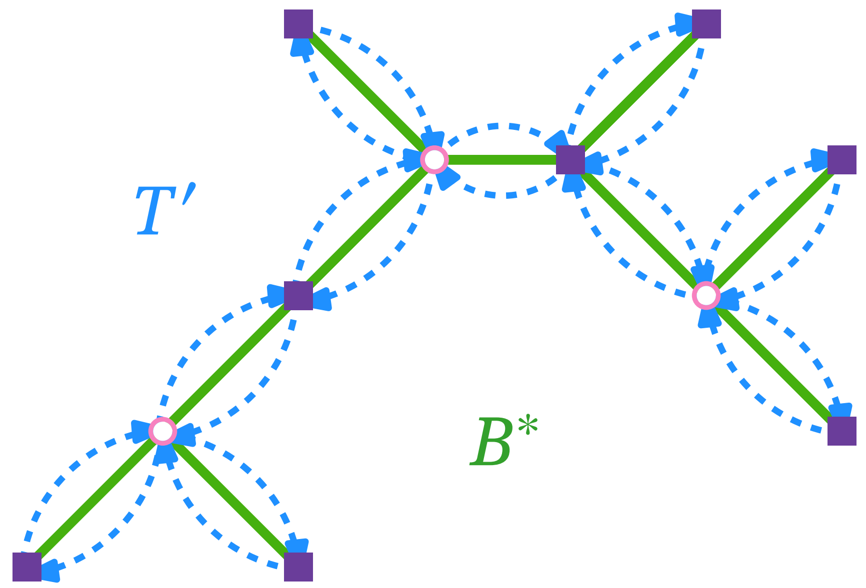
Proof of Approximation Factor

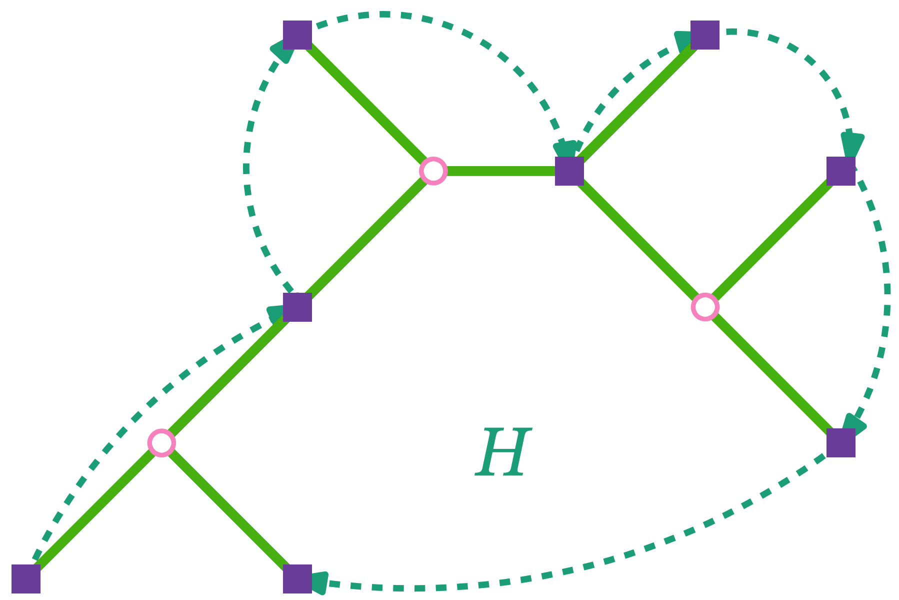
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by "short-cutting" Steiner vertices and already visited terminals.
$\Rightarrow c(H) \leq c(T') = 2 \cdot \mathrm{OPT}$ since G is metric.





Proof of Approximation Factor

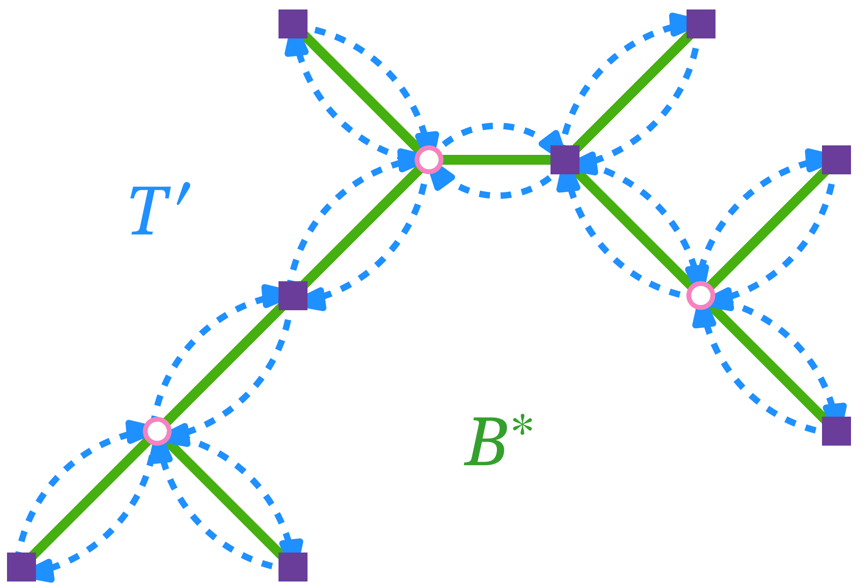
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

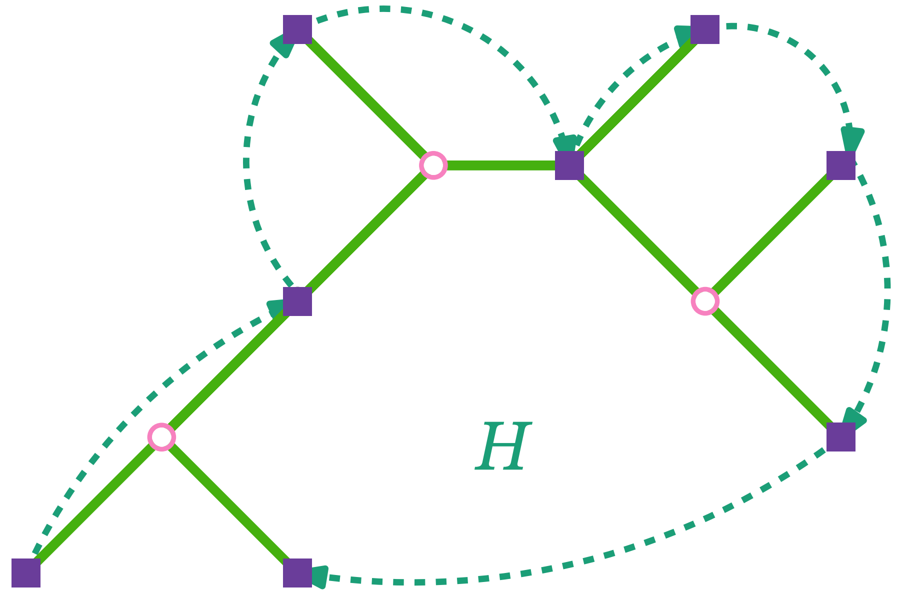
Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by "short-cutting" Steiner vertices and already visited terminals.
$\Rightarrow c(H) \leq c(T') = 2 \cdot \mathrm{OPT}$ since G is metric.

MST B of $G[T]$ has cost $c(B) \leq c(H)$





Proof of Approximation Factor

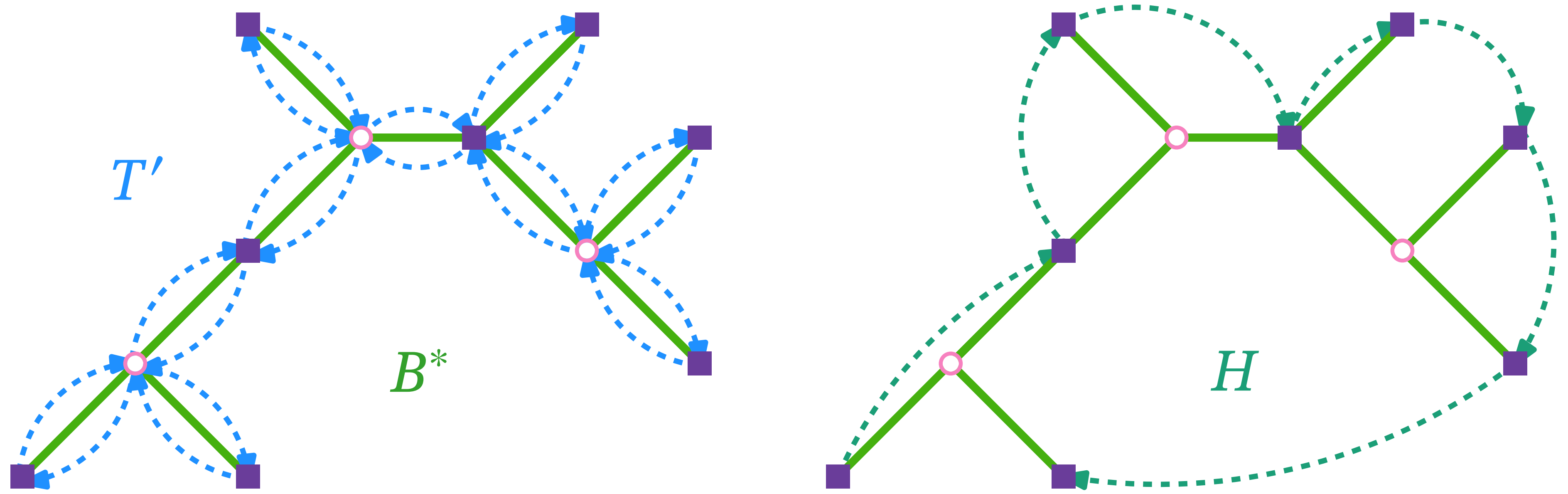
Consider an optimal Steiner tree B^* i.e., $c(B^*) = \mathrm{OPT}$.

Duplicate all edges of B^* $\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \mathrm{OPT}$.

Find a Eulerian tour T' in B' $\quad \Rightarrow c(T') = c(B') = 2 \cdot \mathrm{OPT}$

Find a Hamiltonian path H in $G[T]$ by "short-cutting" Steiner vertices and already visited terminals.
$\Rightarrow c(H) \leq c(T') = 2 \cdot \mathrm{OPT}$ since G is metric.

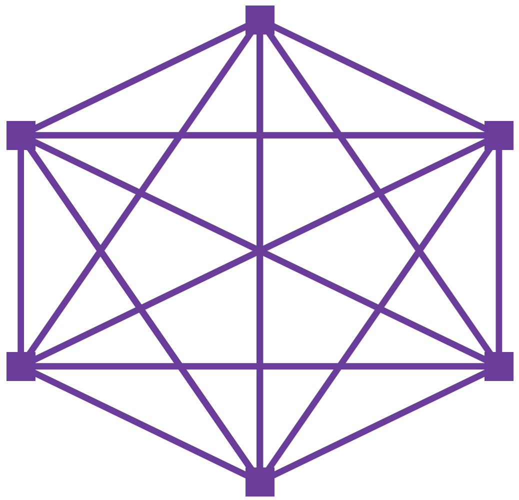
MST B of $G[T]$ has cost $c(B) \leq c(H) \leq 2 \cdot \mathrm{OPT}$ since H is a spanning tree of $G[T]$.



Analysis Tight?

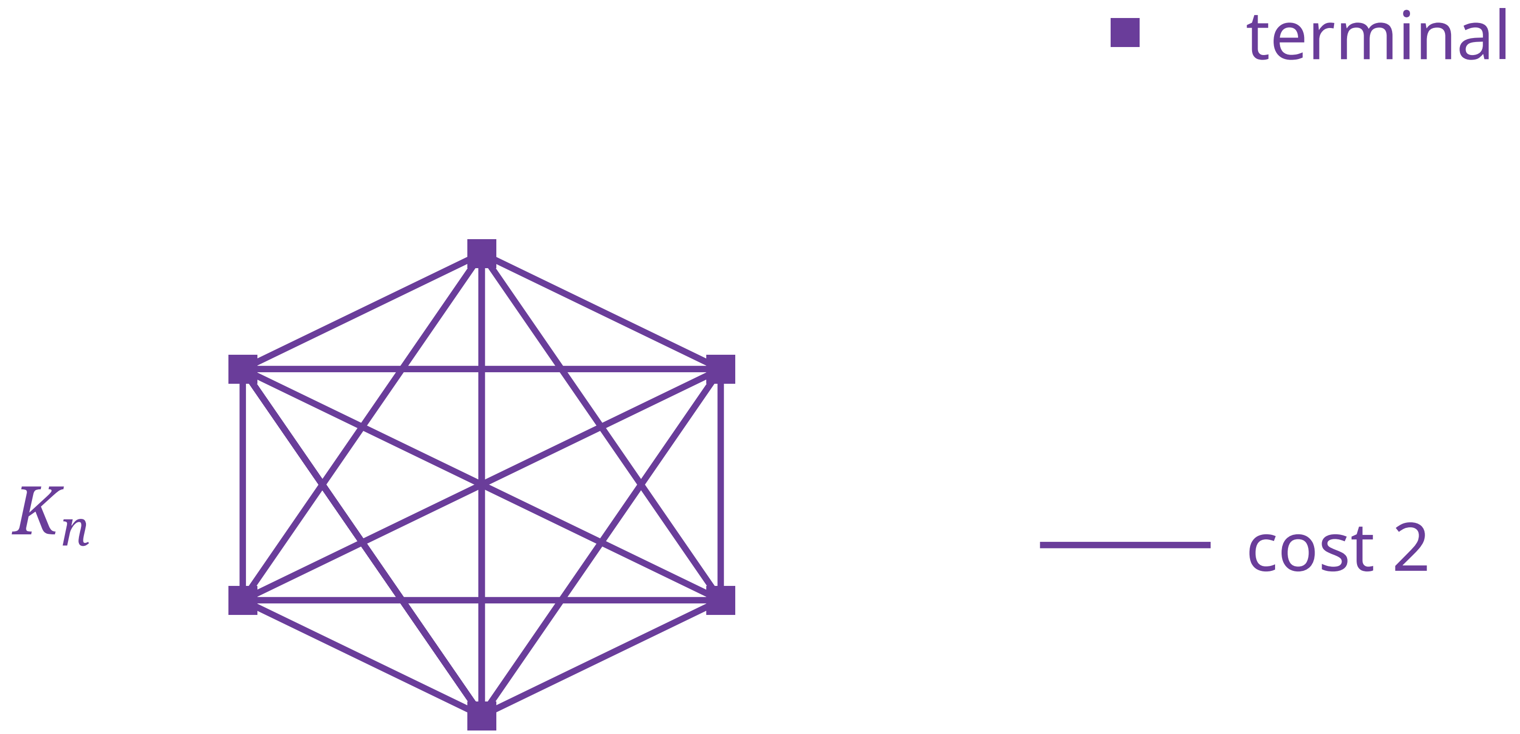
Analysis Tight?

K_n

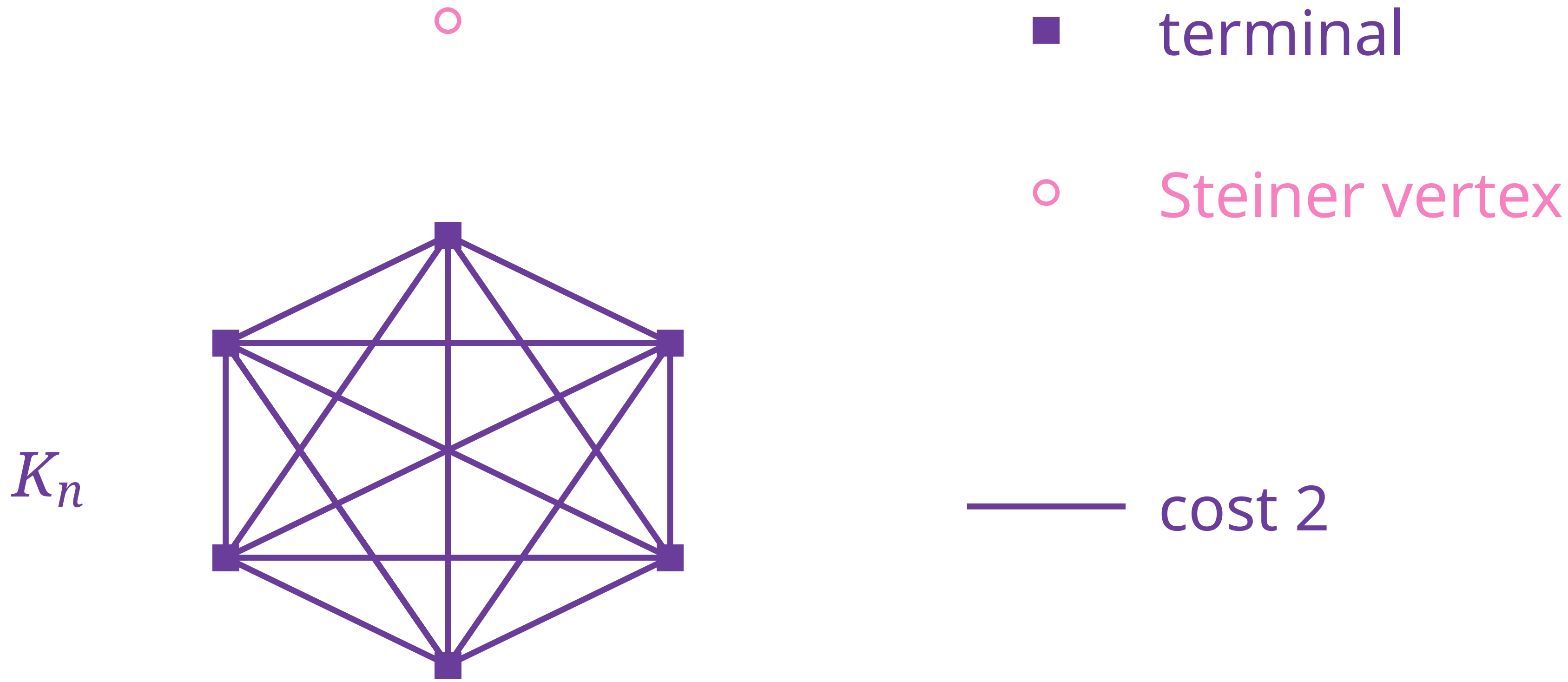


■ terminal

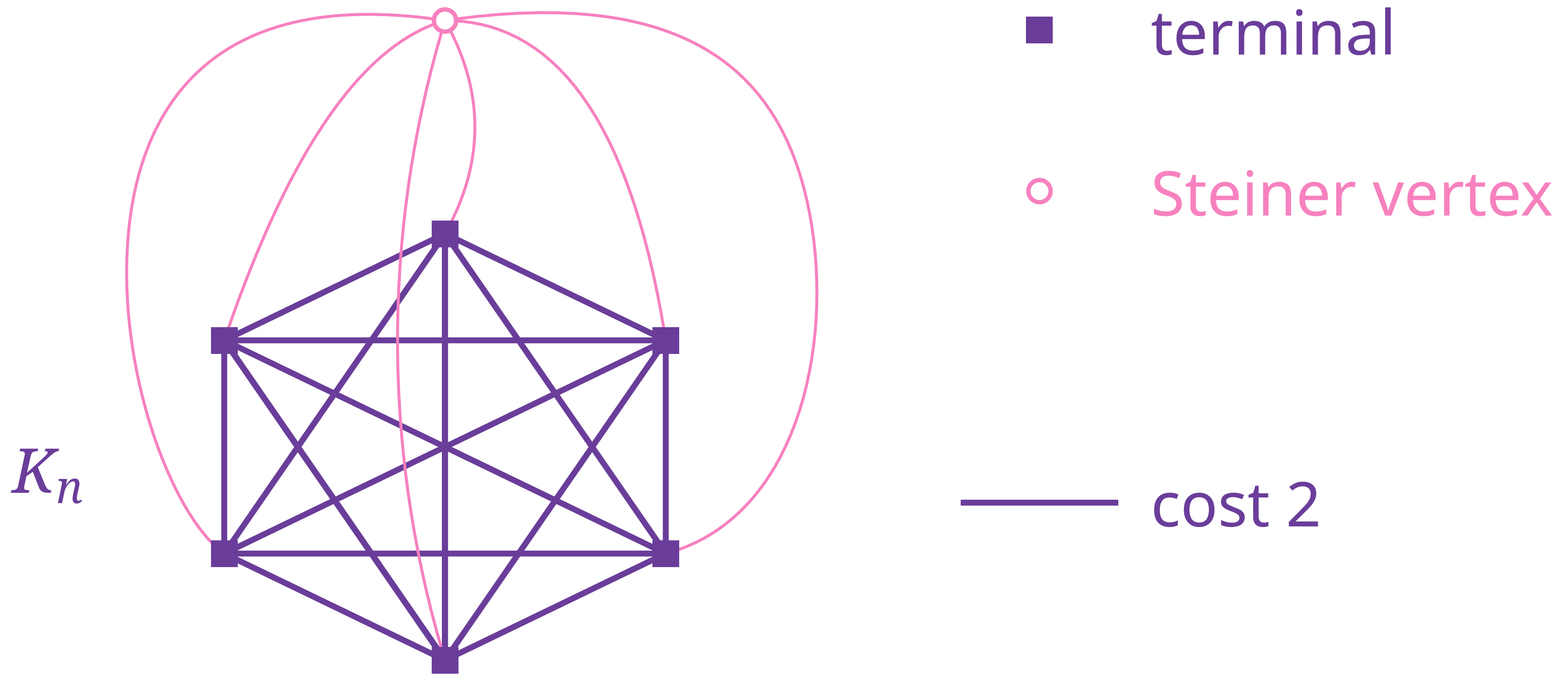
Analysis Tight?



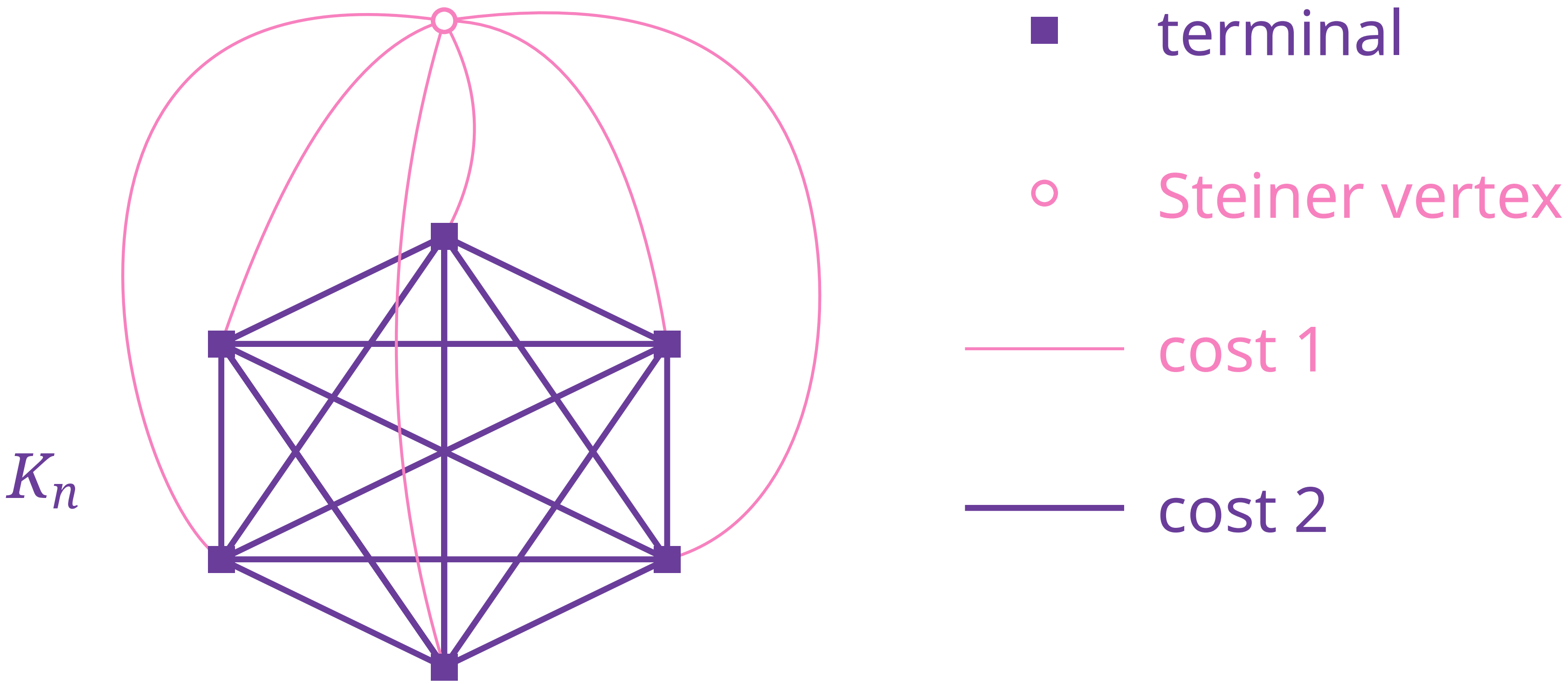
Analysis Tight?



Analysis Tight?

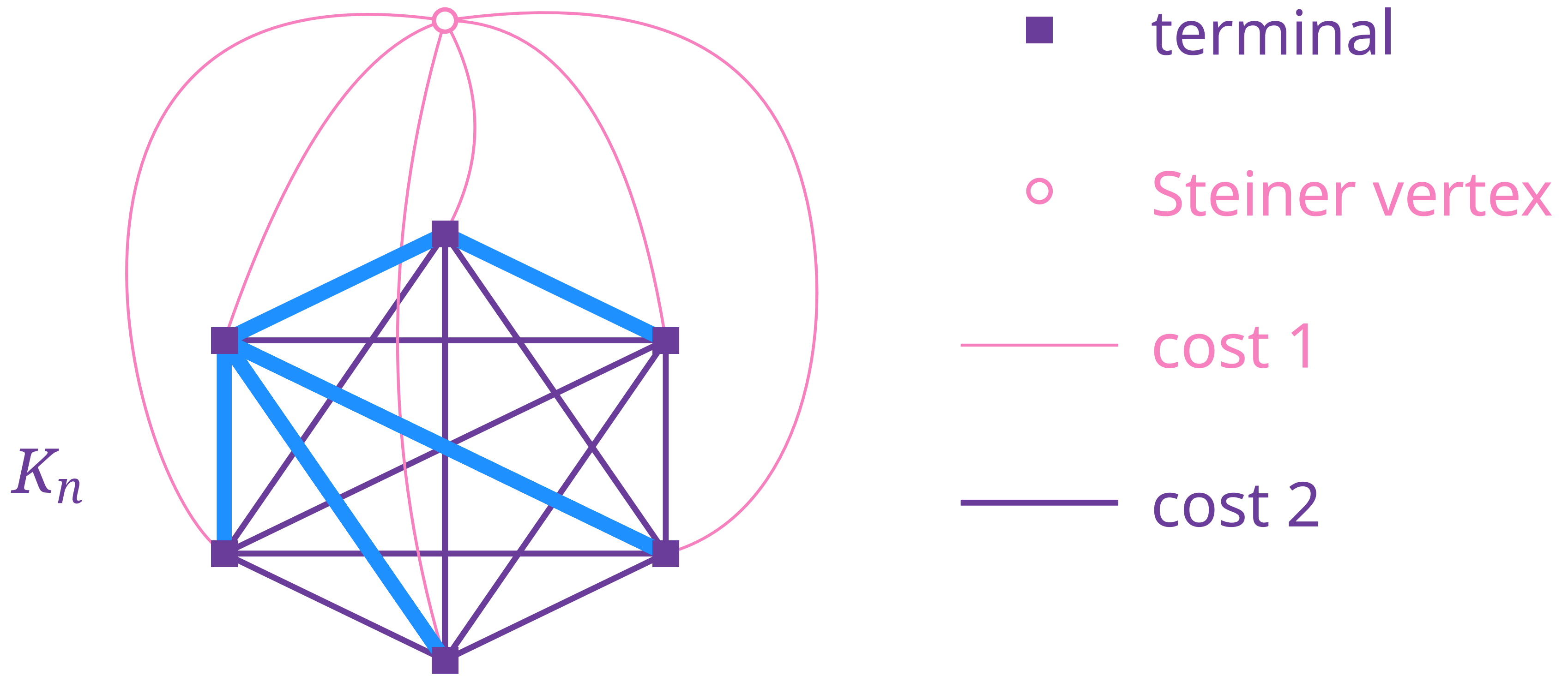


Analysis Tight?



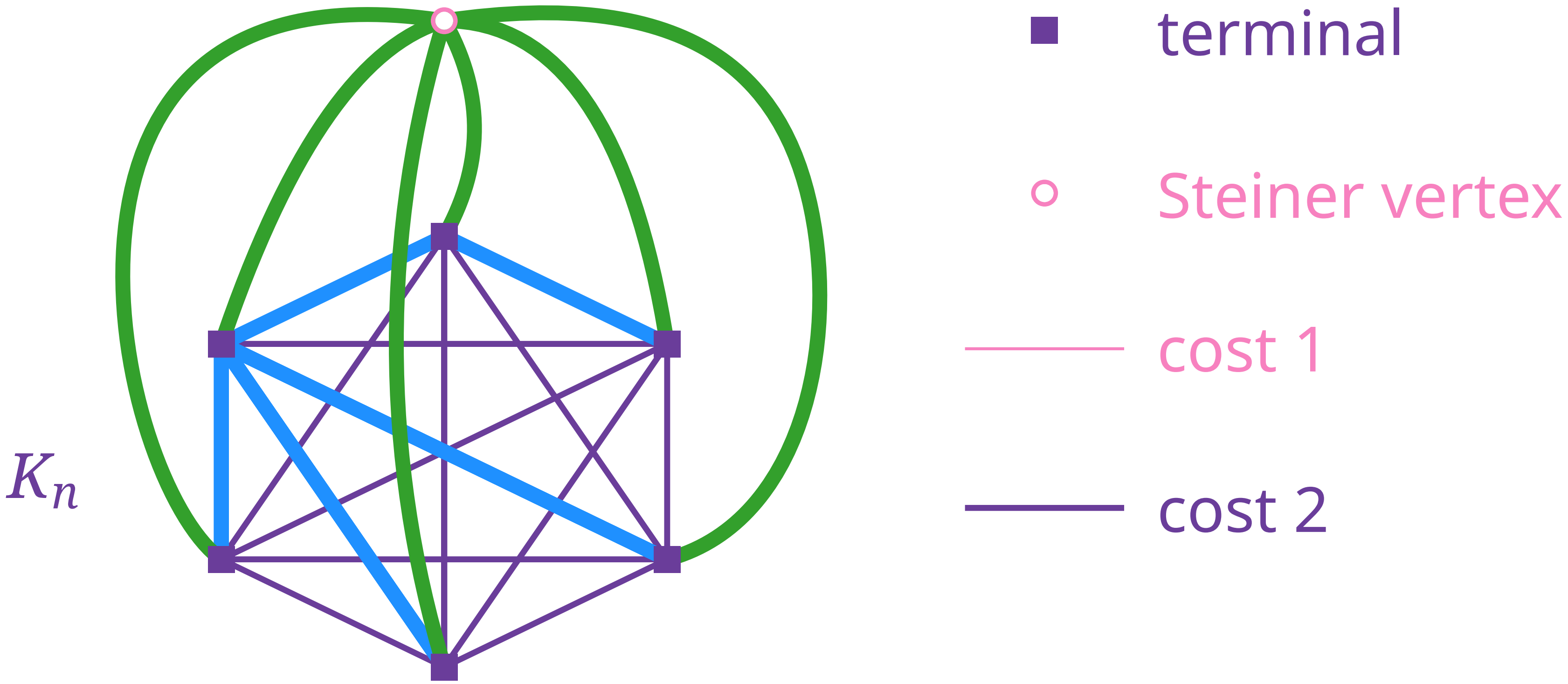
Analysis Tight?

MST of $G[T]$ with cost $2(n-1)$



Analysis Tight?

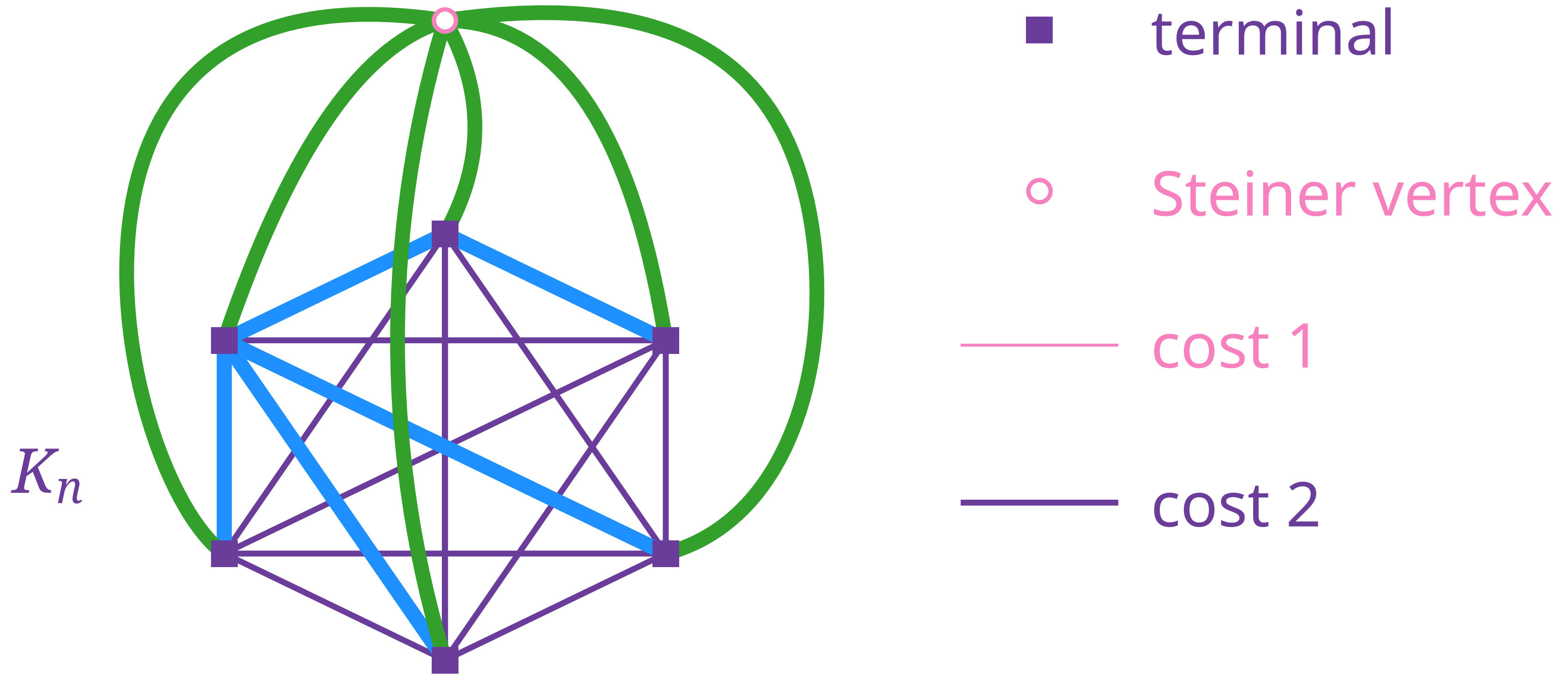
MST of $G[T]$ with cost $2(n-1)$
Optimal solution with cost n



Analysis Tight?

MST of $G[T]$ with cost $2(n-1)$
Optimal solution with cost n

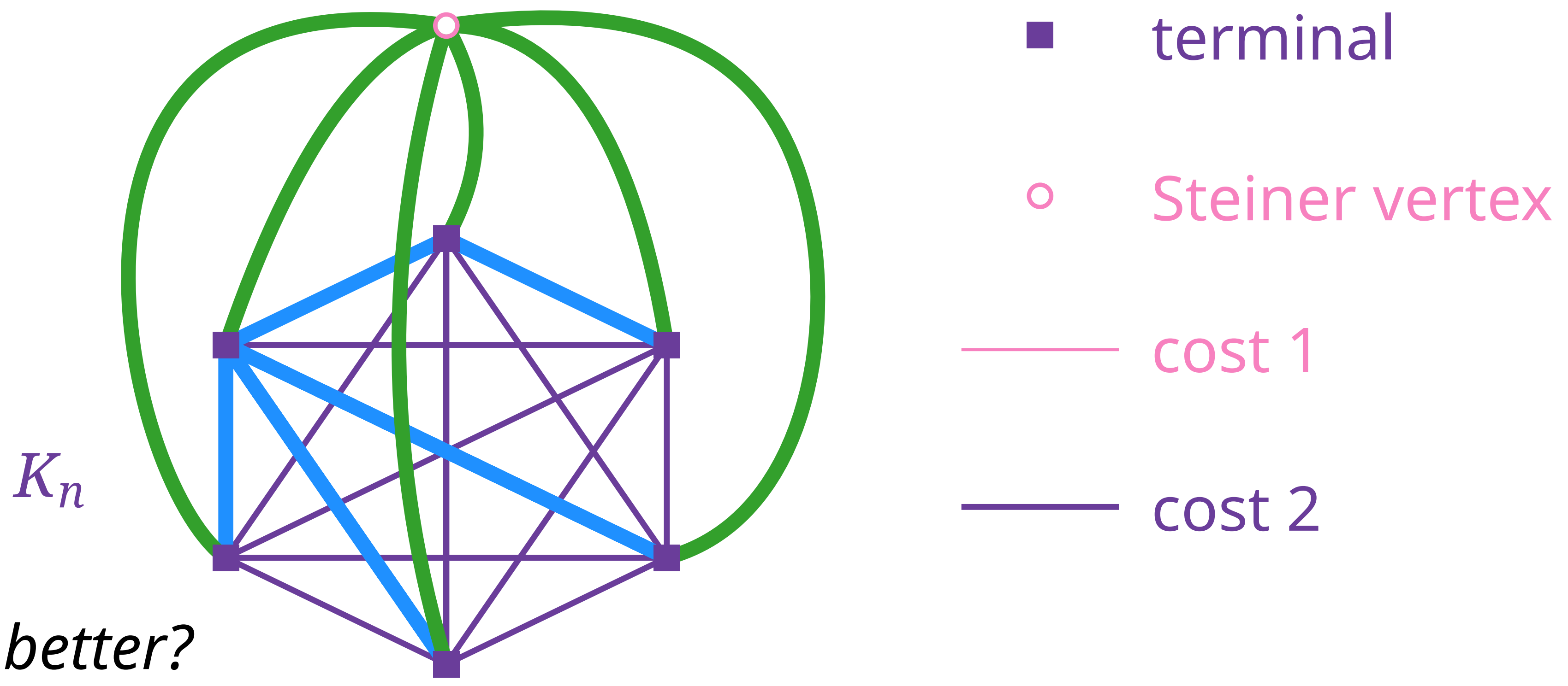
$$\frac{2(n-1)}{n} \rightarrow 2$$



Analysis Tight?

MST of $G[T]$ with cost $2(n-1)$
Optimal solution with cost n

$$\frac{2(n-1)}{n} \to 2$$

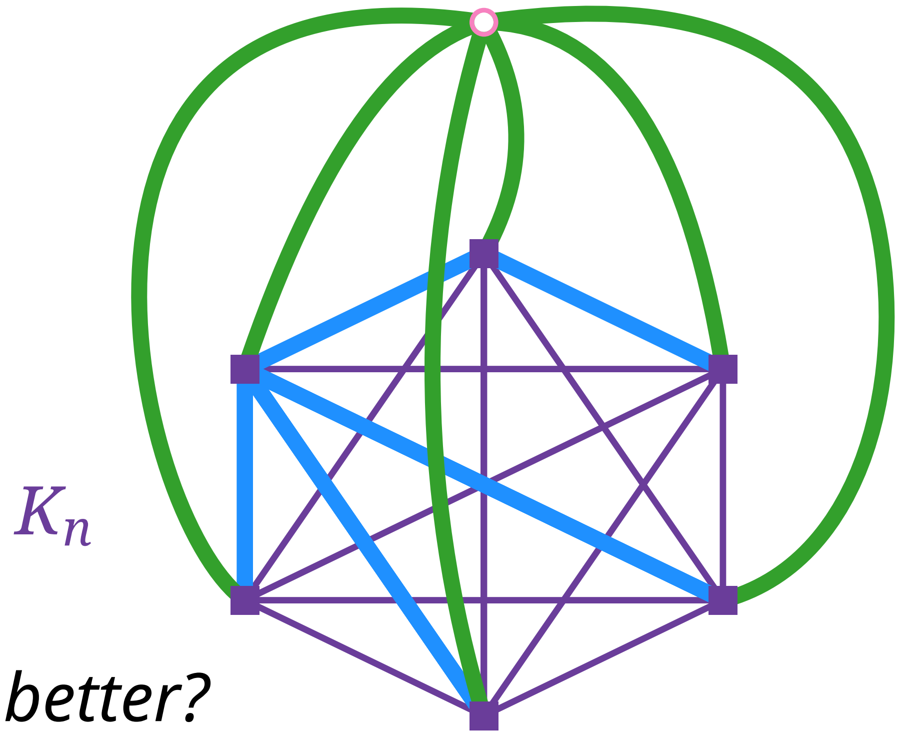


Can we do better?

Analysis Tight?

MST of $G[T]$ with cost $2(n-1)$
Optimal solution with cost n

$$\frac{2(n-1)}{n} \to 2$$



terminal

Steiner vertex

cost 1

cost 2

Can we do better?

The best known approximation factor for STEINERTREE is $\ln(4) + \varepsilon \approx 1.39$

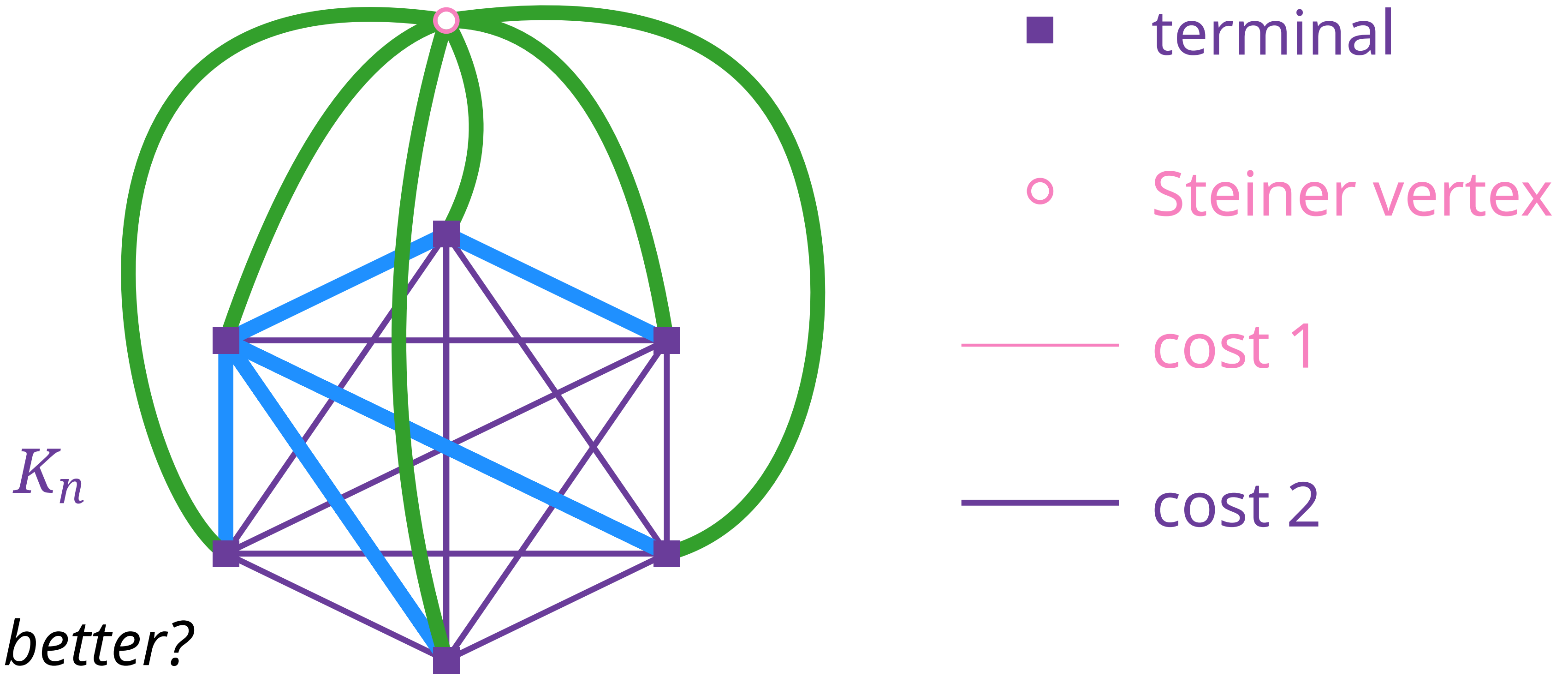
[Byrka, Grandoni, Rothvoß & Sanità, J. ACM'13]

Analysis Tight?

MST of $G[T]$ with cost $2(n-1)$

Optimal solution with cost n

$$\frac{2(n-1)}{n} \to 2$$



Can we do better?

The best known approximation factor for STEINERTREE is $\ln(4) + \varepsilon \approx 1.39$

[Byrka, Grandoni, Rothvoß & Sanità, J. ACM'13]

STEINERTREE cannot be approximated within factor $\frac{96}{95} \approx 1.0105$ (unless P=NP)

[Chlebík & Chlebíková, TCS'08]

Traveling Salesperson Problem

Traveling Salesperson Problem

Given: A complete graph G with edge weights $c \colon E(G) \to \mathbb{Q}^+$

Find: A Hamilton cycle C of G of minimum cost $c(C) := \sum_{e \in E(C)} c(e)$.

Can we use similar ideas for TSP?

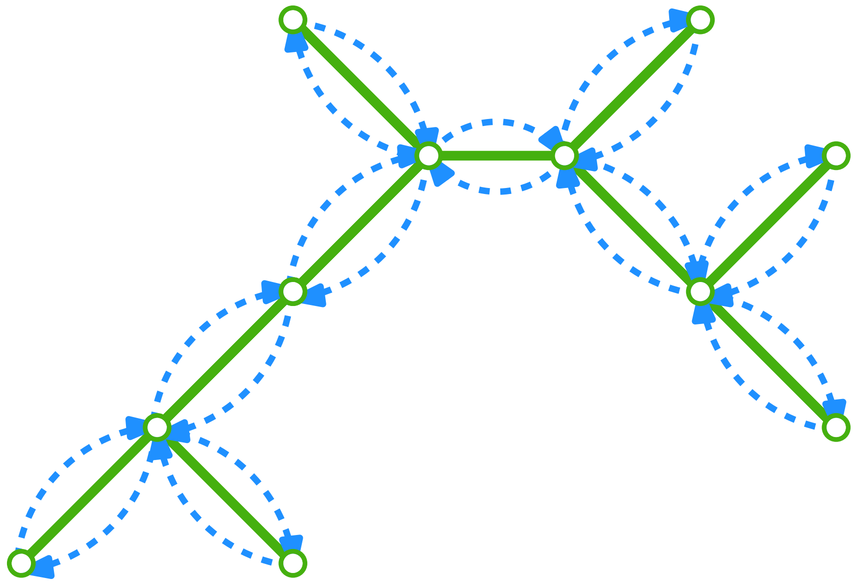
Traveling Salesperson Problem

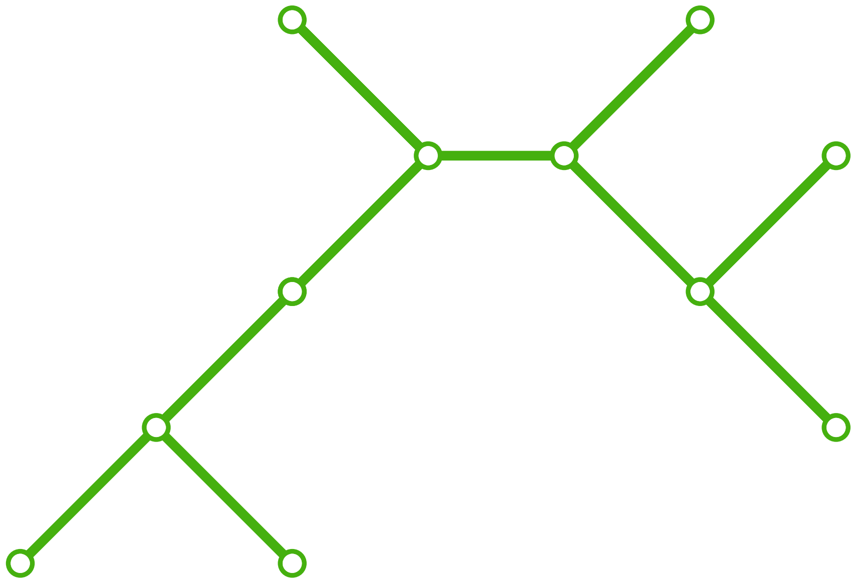
Given: A complete graph G with edge weights $c\colon E(G) \to \mathbb{Q}^{+}$

Find: A Hamilton cycle C of G of minimum cost $c(C) := \sum_{e \in E(C)} c(e)$.

Can we use similar ideas for TSP?

Yes, and we nearly already saw how to construct a TSP from an MST.





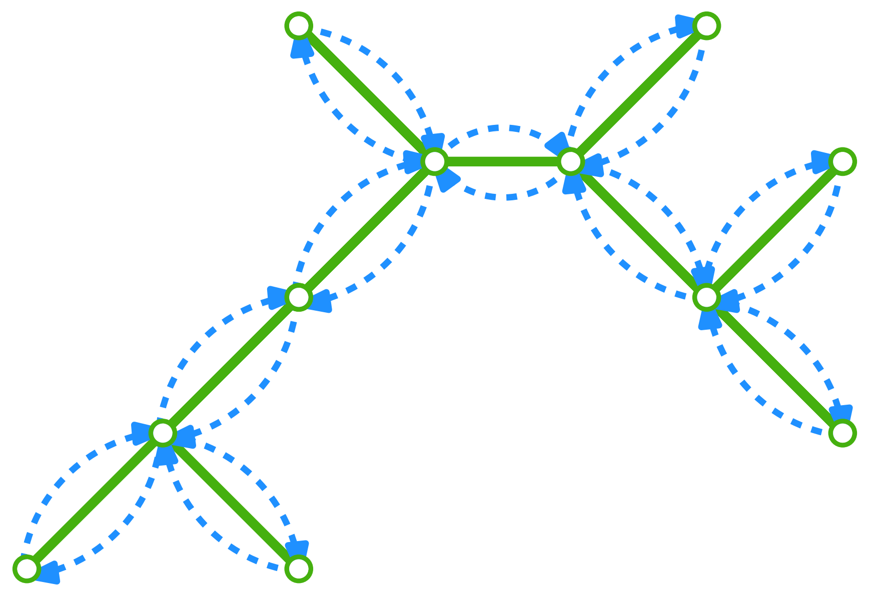
Traveling Salesperson Problem

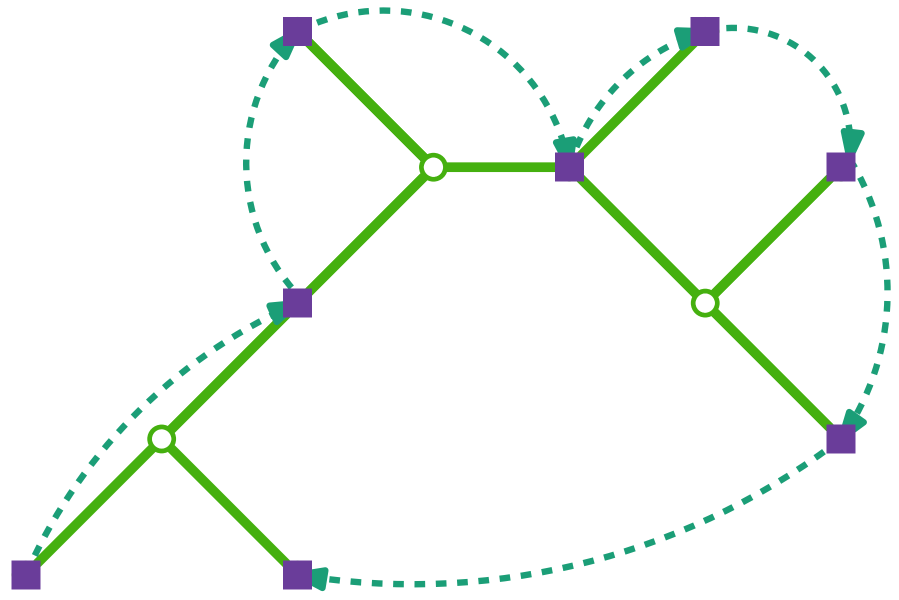
Given: A complete graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$

Find: A Hamilton cycle C of G of minimum cost $c(C) := \sum_{e \in E(C)} c(e)$.

Can we use similar ideas for TSP?

Yes, and we nearly already saw how to construct a TSP from an MST.





For SteinerTree:
skip non-terminals
and repeated vertices

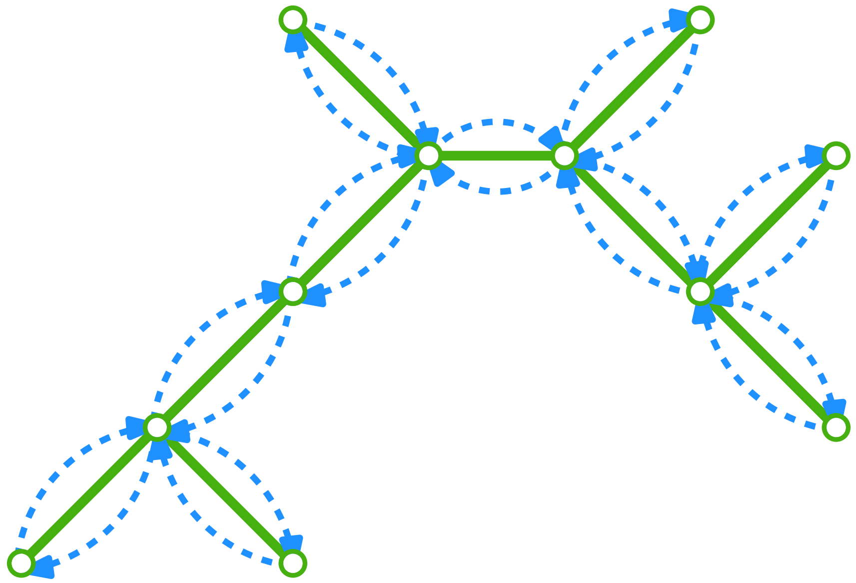
Traveling Salesperson Problem

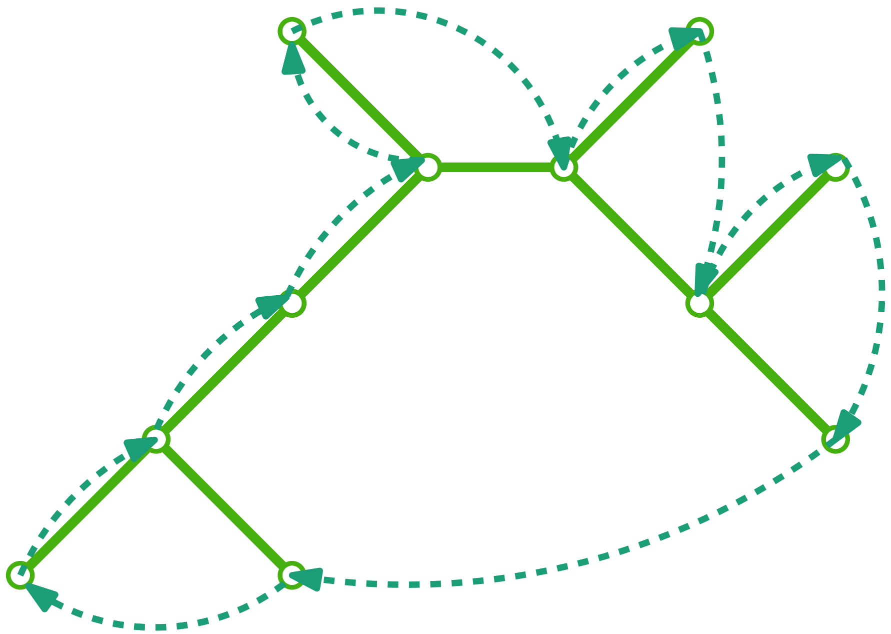
Given: A complete graph G with edge weights $c\colon E(G) \to \mathbb{Q}^{+}$

Find: A Hamilton cycle C of G of minimum cost $c(C) := \sum_{e \in E(C)} c(e)$.

Can we use similar ideas for TSP?

Yes, and we nearly already saw how to construct a TSP from an MST.





For TSP:
skip repeated vertices
(except for last one)

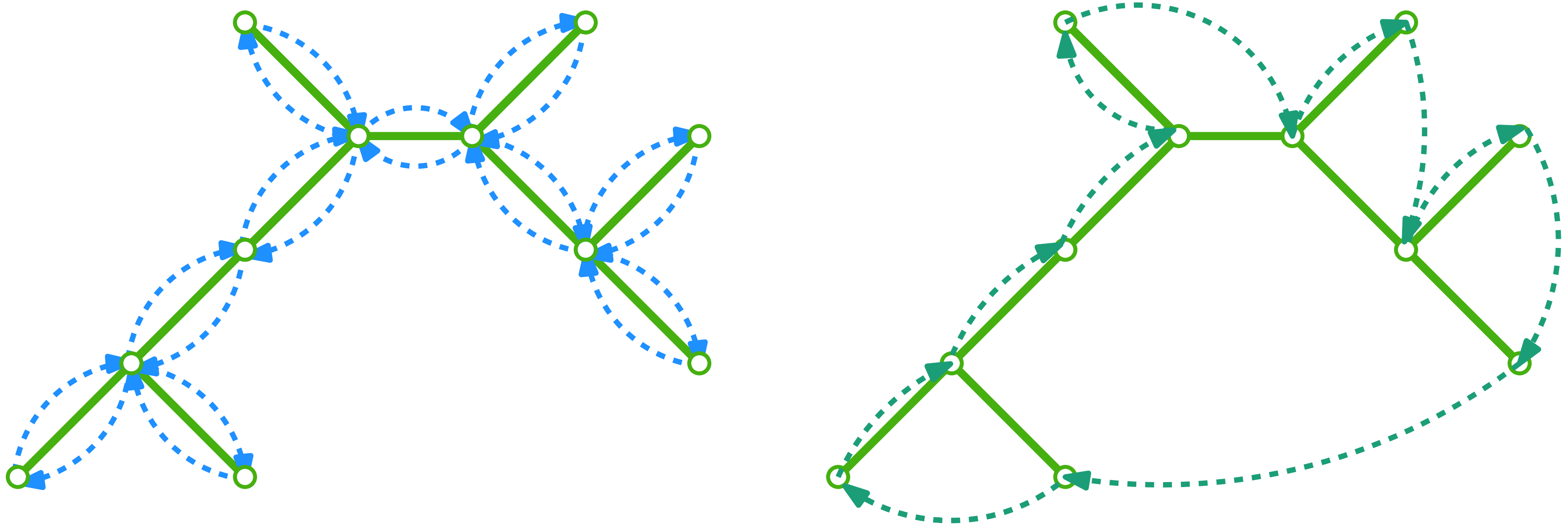
Traveling Salesperson Problem

Given: A complete graph G with edge weights $c\colon E(G) \to \mathbb{Q}^+$

Find: A Hamilton cycle C of G of minimum cost $c(C) := \sum_{e \in E(C)} c(e)$.

Can we use similar ideas for TSP?

Yes, and we nearly already saw how to construct a TSP from an MST.



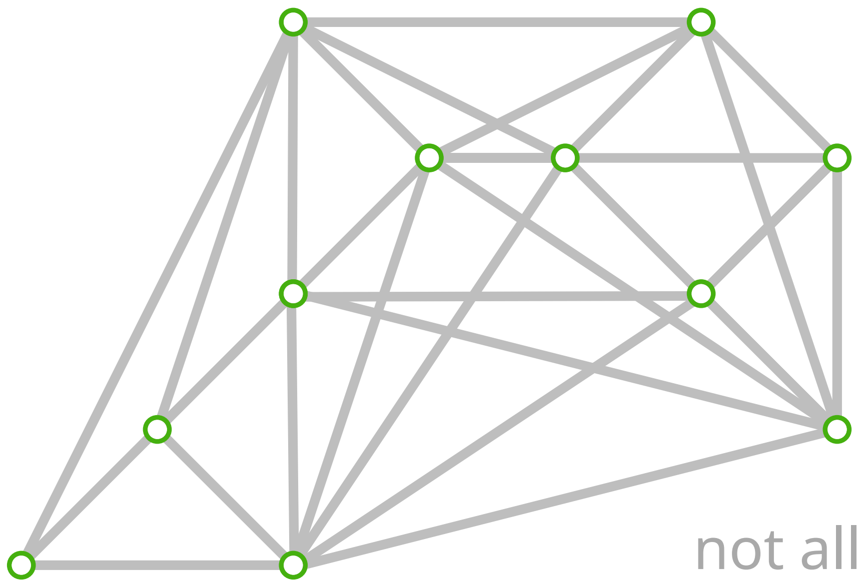
For TSP:
skip repeated vertices
(except for last one)

But this will only work for metric TSP (non-metric TSP is hard to approximate)

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

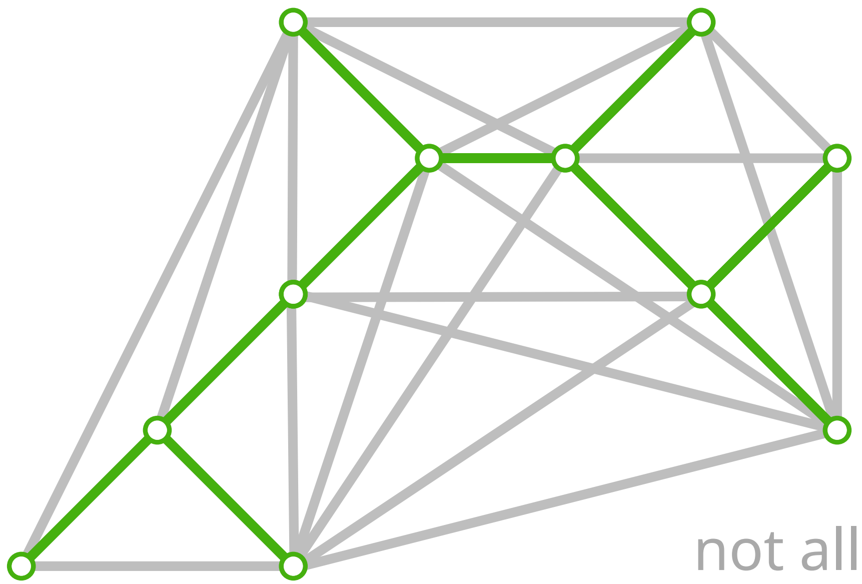


not all edges in complete graph shown

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

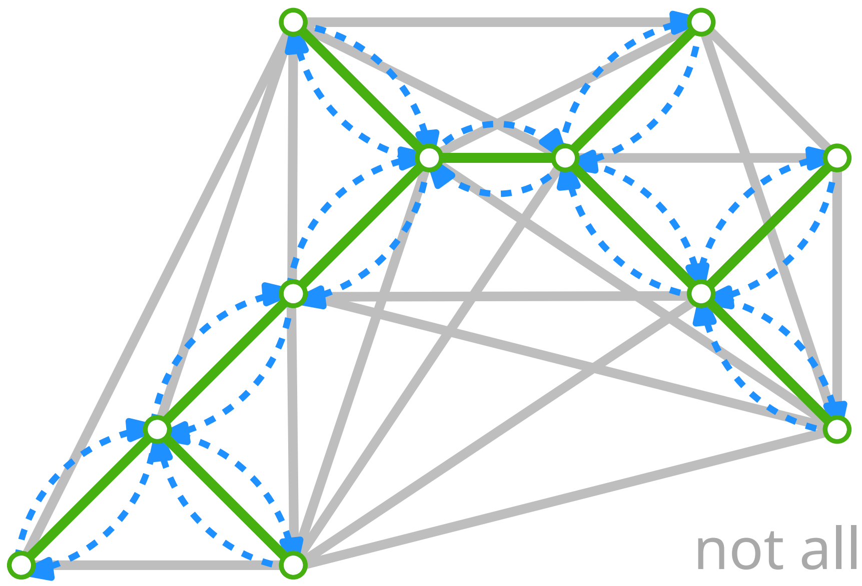


not all edges in complete graph shown

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

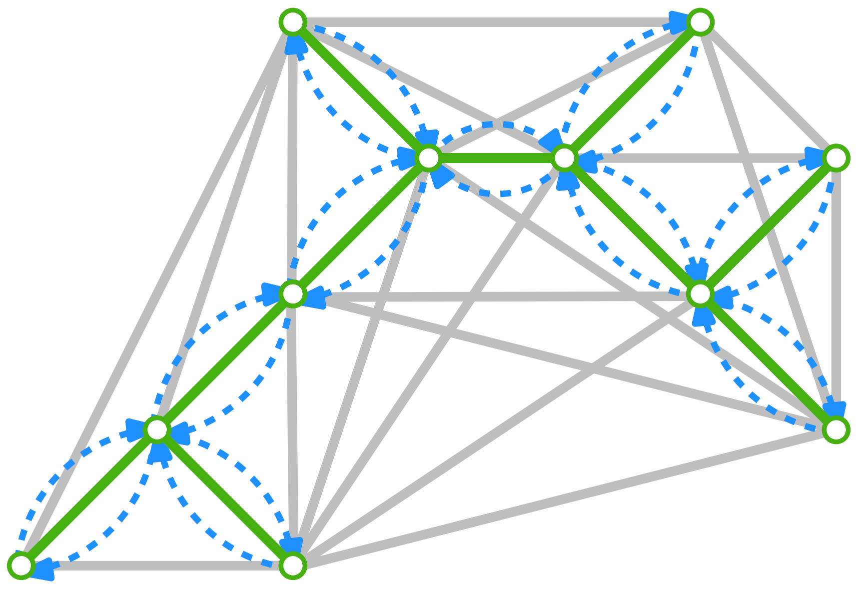


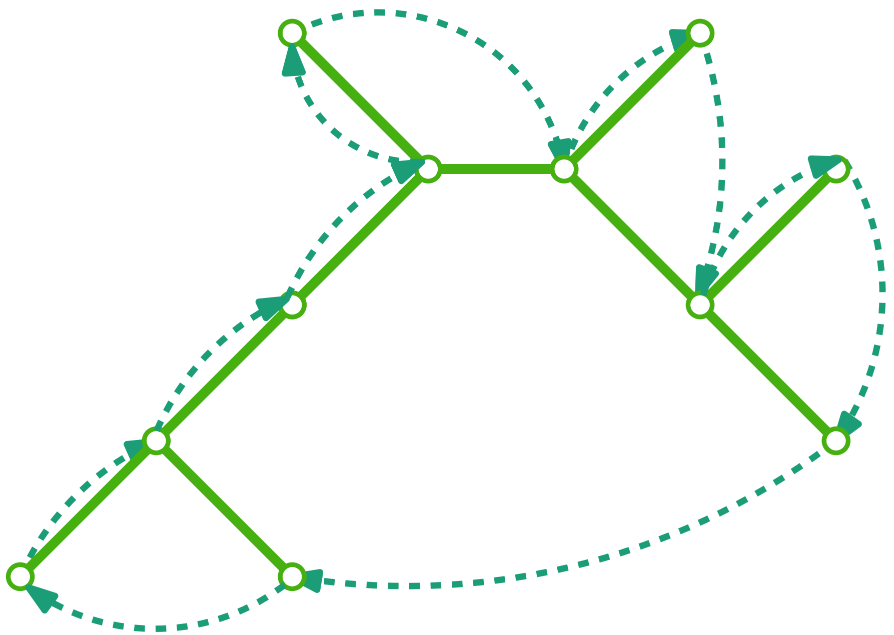
not all edges in complete graph shown

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C



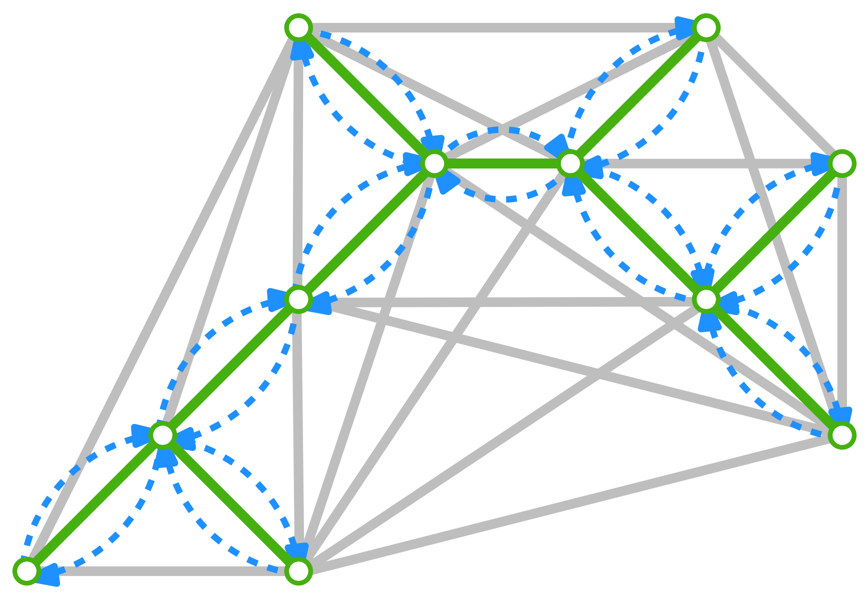


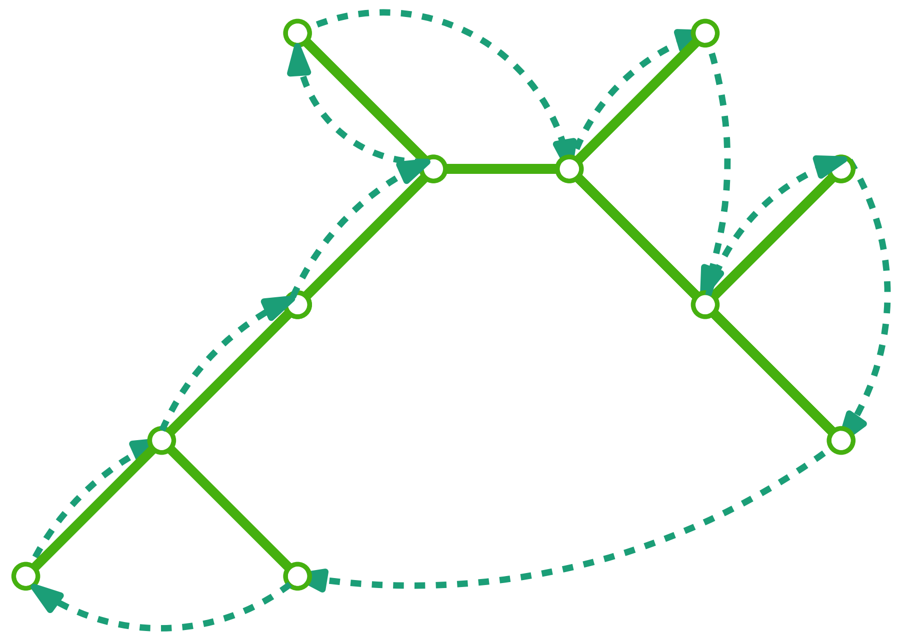
Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.





Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:
- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Proof of Approximation factor:

$\text{cost}(T) \leq \text{OPT}$ removing one edge from a TSP tour gives a spanning tree

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Proof of Approximation factor:

$\text{cost}(T) \leq \text{OPT}$ removing one edge from a TSP tour gives a spanning tree

$$\text{cost}(E) = 2\text{cost}(T) \leq 2\text{OPT}$$

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Proof of Approximation factor:

$\text{cost}(T) \leq \text{OPT}$ removing one edge from a TSP tour gives a spanning tree

$\text{cost}(C) \leq \text{cost}(E) = 2\text{cost}(T) \leq 2\text{OPT}$

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for MetricTSP.

Is the analysis tight?

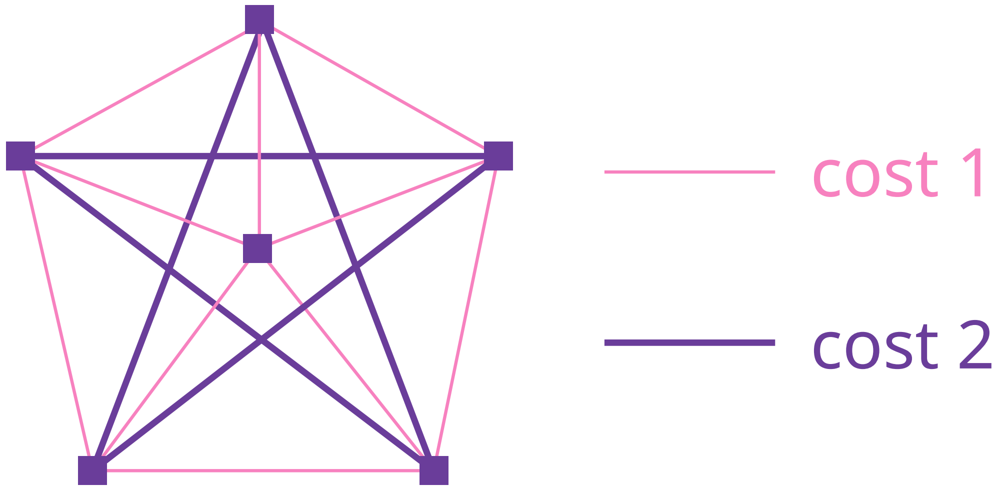
Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:
- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Is the analysis tight? Yes!

K_n



Metric Traveling Salesperson Problem (Metric TSP)

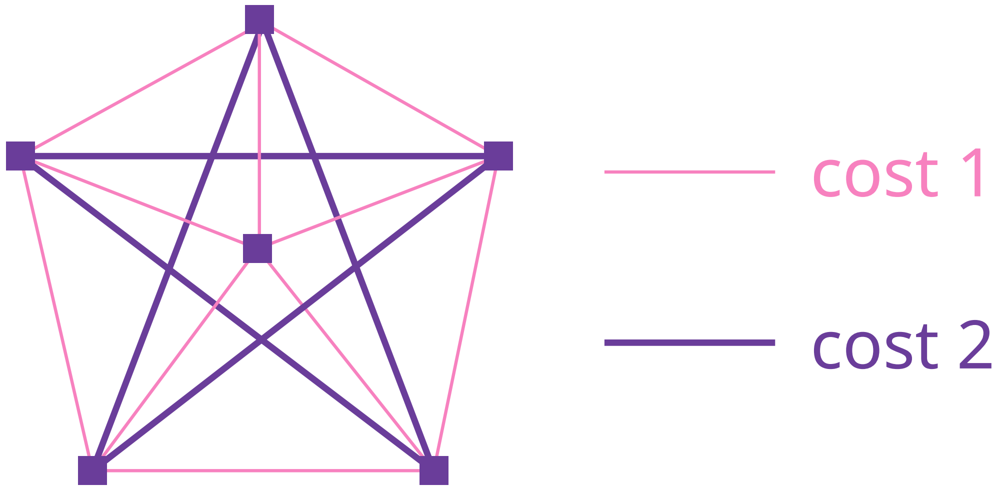
Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Is the analysis tight? Yes!

K_n



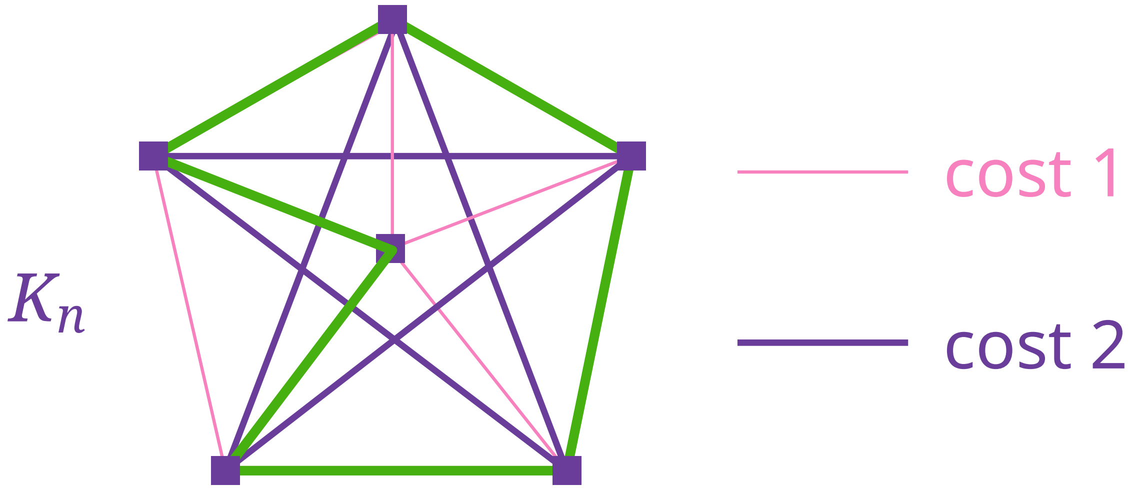
What is OPT?

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:
- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Is the analysis tight? Yes!



What is OPT? n

Metric Traveling Salesperson Problem (Metric TSP)

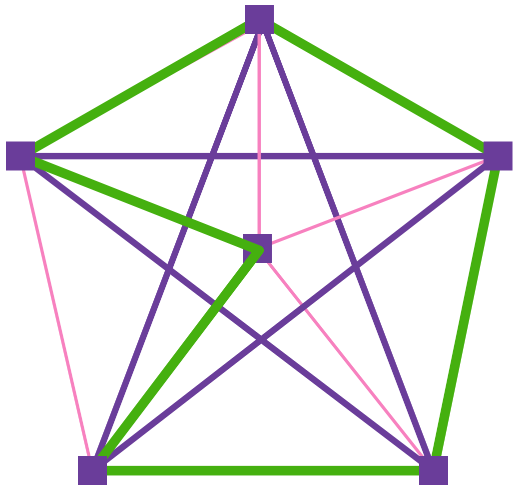
Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for METRICTSP.

Is the analysis tight? Yes!

K_n



cost 1

cost 2

What is OPT? n

What does the algorithm compute?

Metric Traveling Salesperson Problem (Metric TSP)

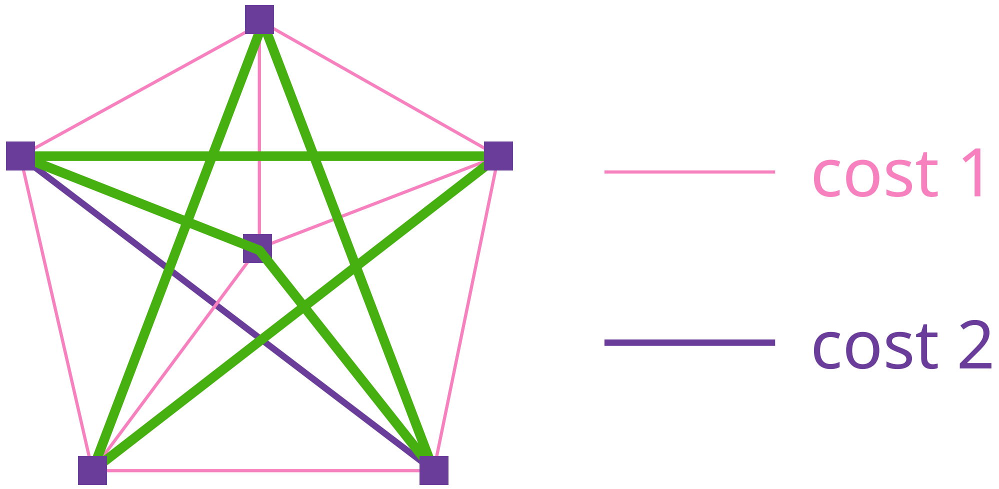
Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for MetricTSP.

Is the analysis tight? Yes!

K_n



What is OPT? n

What does the algorithm compute?

depends on Eulertour
but can be $2n - 1$

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for MetricTSP.

Can we improve this algorithm?

Metric Traveling Salesperson Problem (Metric TSP)

Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- double the edges in T to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 2-approximation for MetricTSP.

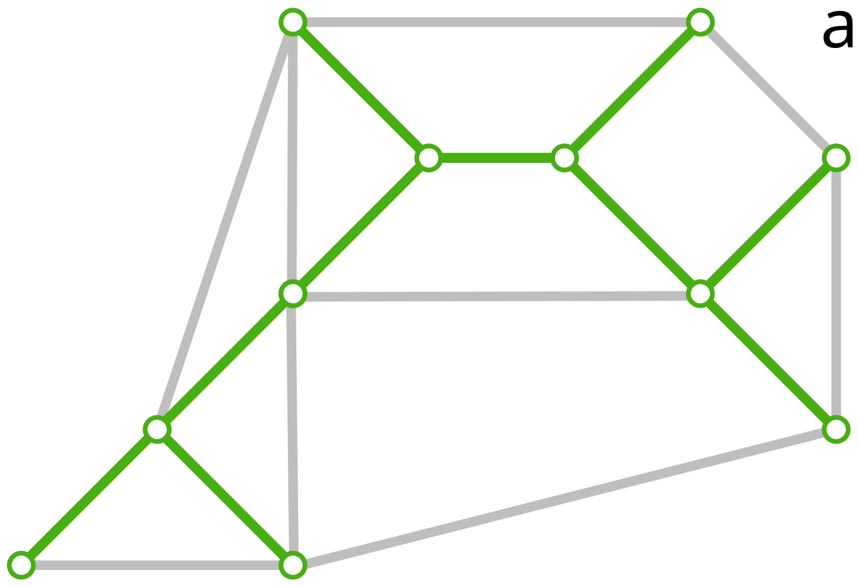
Can we improve this algorithm? Yes!

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- ~~double the edges in T to get T'~~ add mincost matching M to $T \rightarrow T'$
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Can we improve this algorithm? Yes! instead of doubling the edges in T add a mincost perfect matching of all odd degree vertices

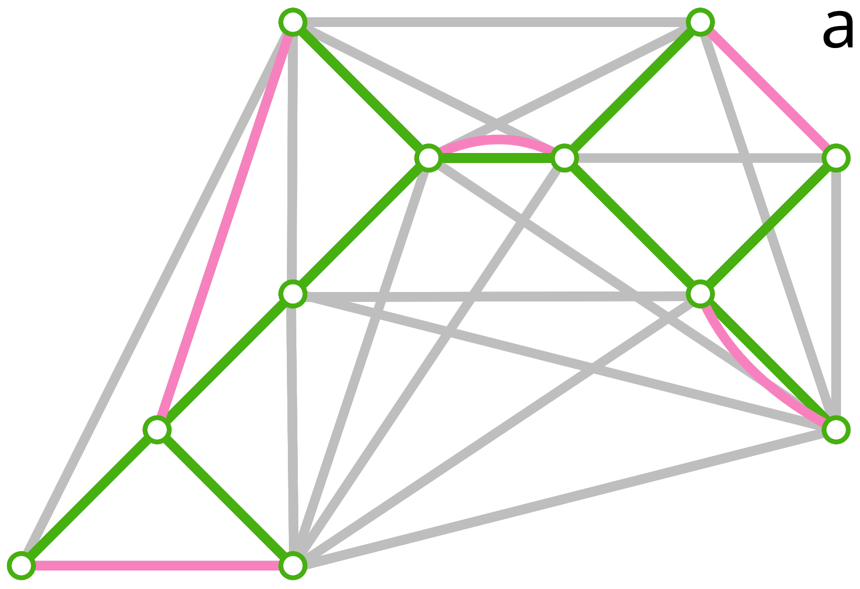


Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Can we improve this algorithm? Yes! instead of doubling the edges in T add a mincost perfect matching of all odd degree vertices

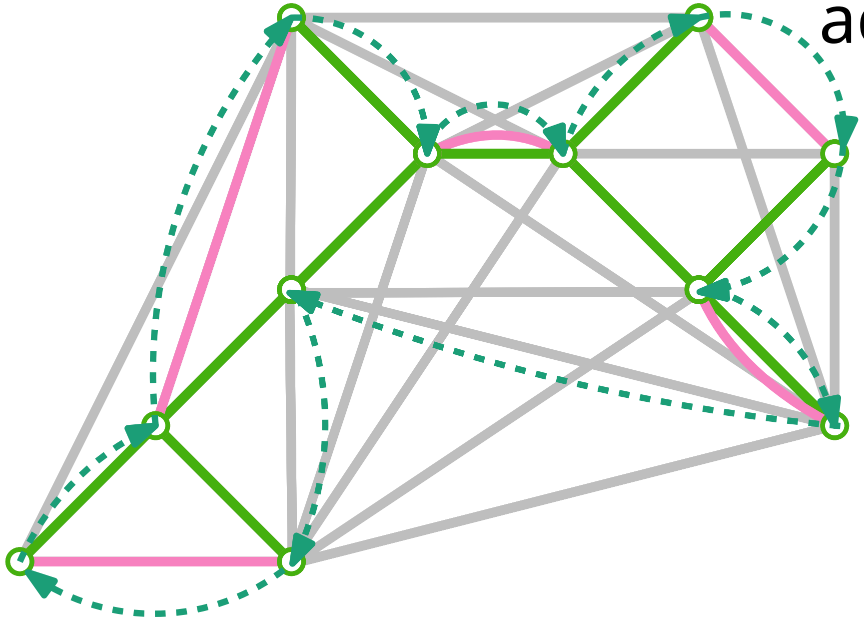


Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Can we improve this algorithm? Yes! instead of doubling the edges in T
add a mincost perfect matching of all odd degree vertices

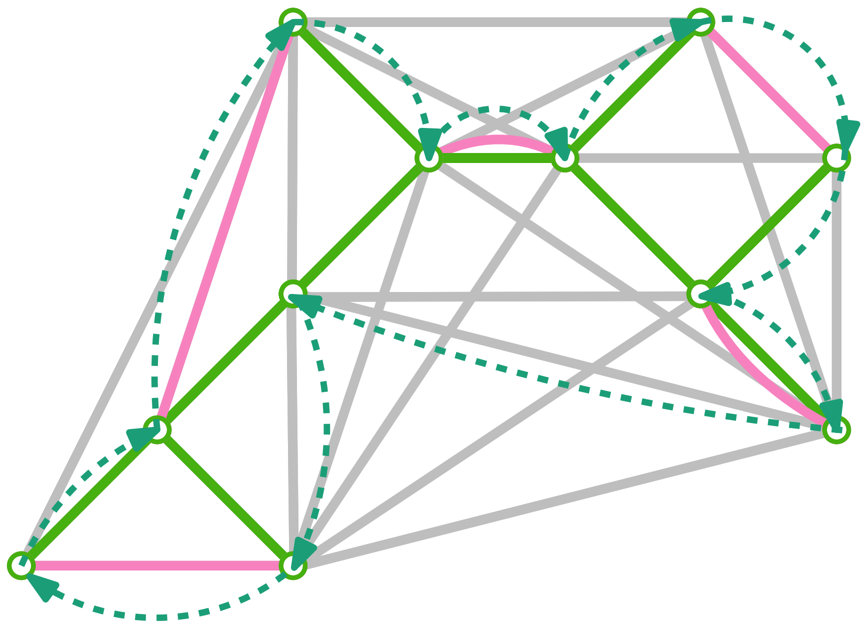


Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.

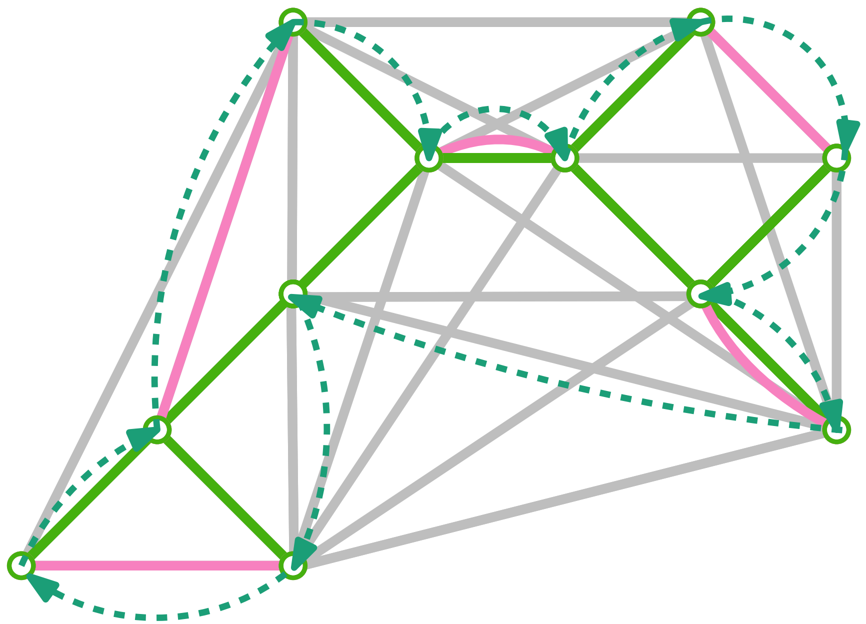


Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.



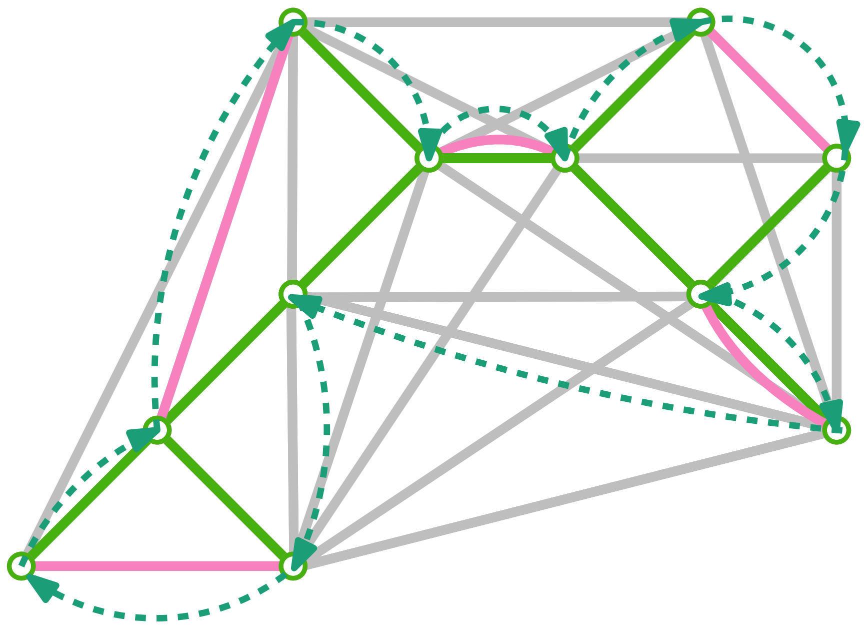
Why does a perfect matching always exist?

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.



Why does a perfect matching always exist?

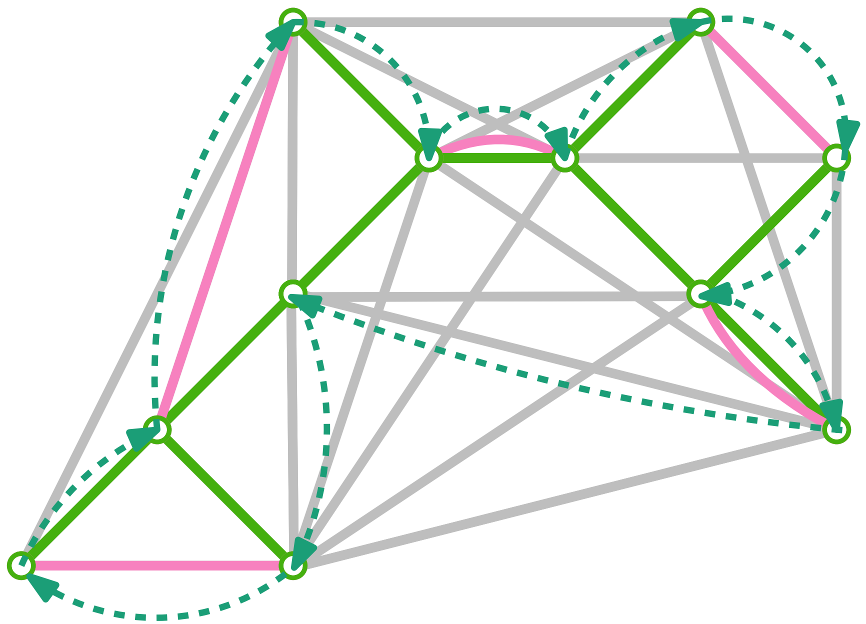
By handshaking lemma the total sum of degrees is even, hence the number of odd degree vertices must be even .

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a 3/2-approximation for TRAVELINGSALESPERSONPROBLEM.



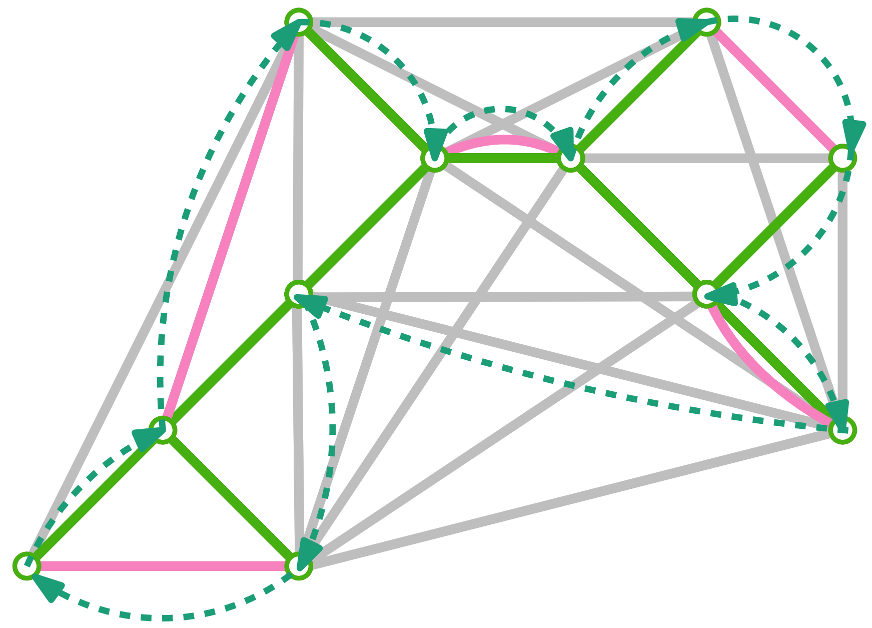
What is the cost of such a mincost matching?

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.



What is the cost of such a mincost matching?

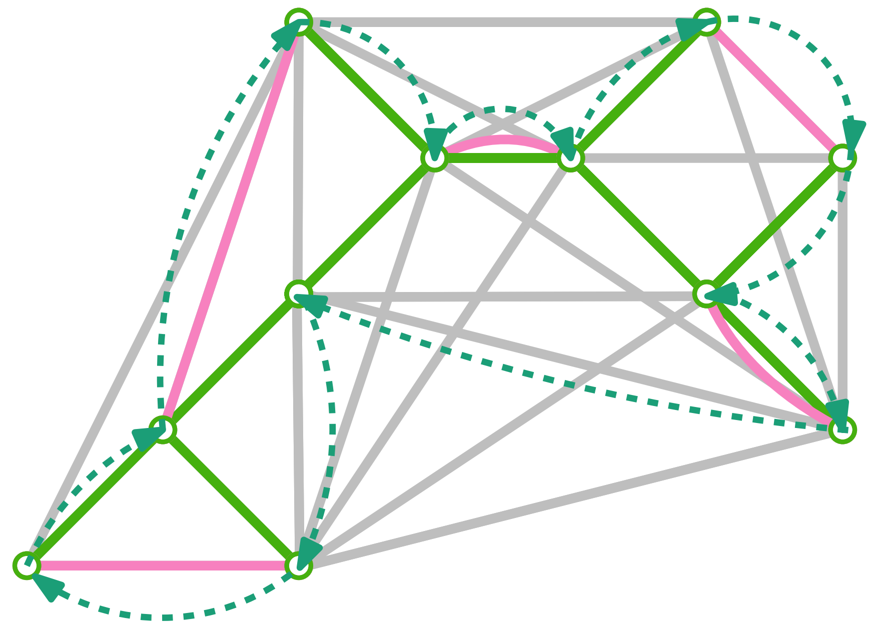
$\text{cost}(M) \leq \text{OPT}/2$ half the cost of a cycle

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.



What is the cost of such a mincost matching?

$\text{cost}(M) \leq \text{OPT}/2$ half the cost of a cycle

Hence the total cost is

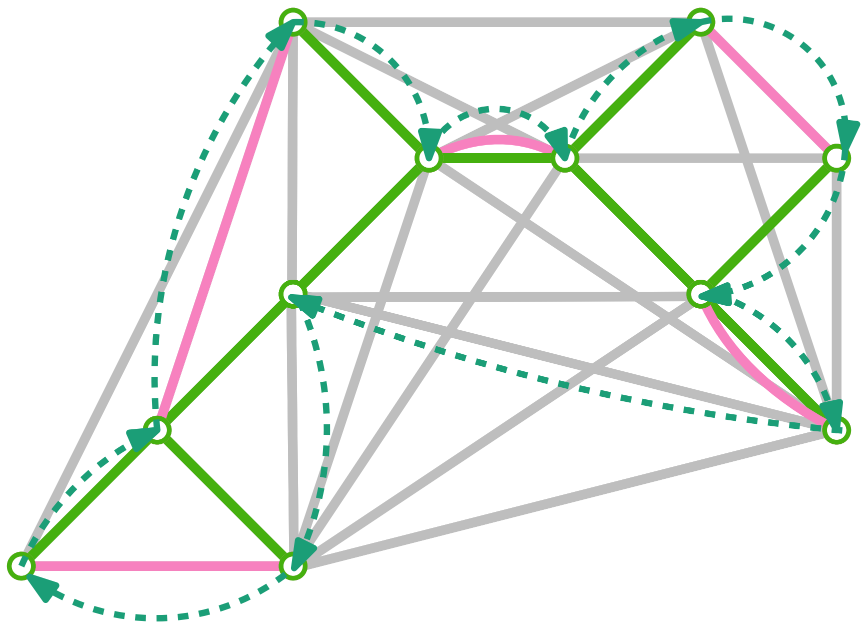
$\text{cost}(C) \leq \text{cost}(E) = \text{cost}(T) + \text{cost}(M) \leq 3/2\ \text{OPT}$

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.



Is the analysis tight?

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

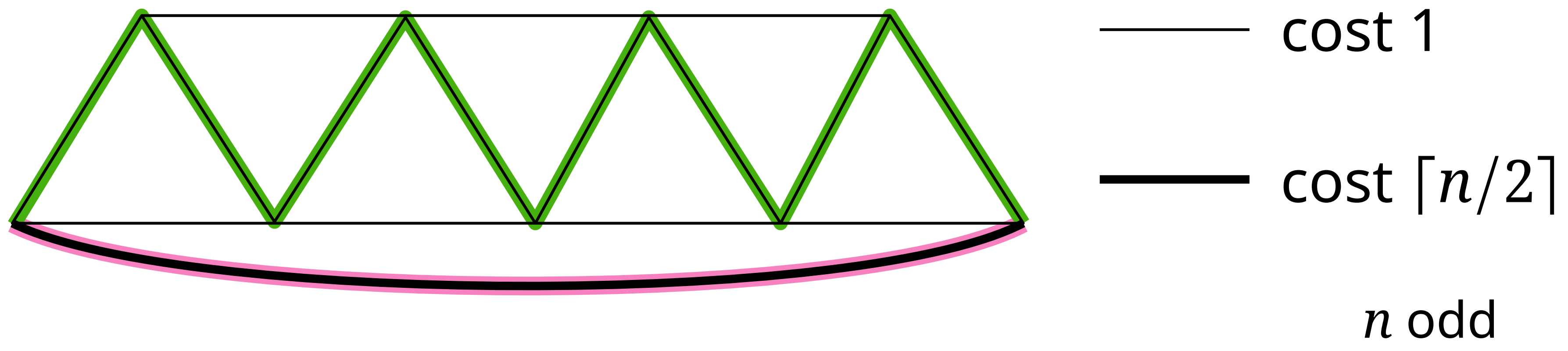
- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.

Is the analysis tight? Yes!

What is OPT?

What does the algorithm compute?



Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

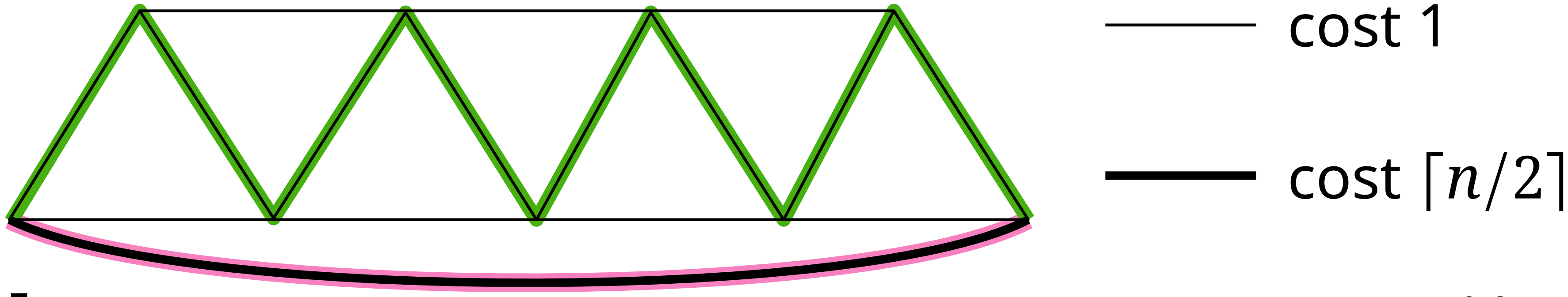
Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.

Is the analysis tight? Yes!

What is OPT? n

What does the algorithm compute?

$n - 1 + \lceil n/2 \rceil$



Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.

Can we do better?

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TravelingSalespersonProblem.

Can we do better?

metric TSP cannot be approximated within $123/122 \approx 1.008$ unless $P = NP$,
and in 2020 the first α-approximation algorithm with $\alpha < 1.5 - 10^{-36}$


ALGORITHMS

Computer Scientists Break Traveling Salesperson Record

24

After 44 years, there's finally a better way to find approximate solutions to the notoriously difficult traveling salesperson problem.

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.

Can we do better?

metric TSP cannot be approximated within $123/122 \approx 1.008$ unless $P = NP$,
and in 2020 the first α-approximation algorithm with $\alpha < 1.5 - 10^{-36}$

for Euclidean TSP there are $(1 + \varepsilon)$-approximation algorithms for any fixed $\varepsilon > 0$
("polynomial-time approximation scheme" (PTAS))

Metric Traveling Salesperson Problem (Metric TSP)

Christofides Algorithm:

- Compute a minimum spanning tree (MST) T in G,
- add mincost perfect matching M on odd degree vertices to get T'
- find an Eulerian tour E in T'
- shortcut repeated vertices in E to get a tour C

Theorem. The algorithm is a $3/2$-approximation for TRAVELINGSALESPERSONPROBLEM.

Can we do better?

metric TSP cannot be approximated within $123/122 \approx 1.008$ unless $P = NP$, and in 2020 the first α-approximation algorithm with $\alpha < 1.5 - 10^{-36}$

for Euclidean TSP there are $(1+\varepsilon)$-approximation algorithms for any fixed $\varepsilon > 0$ ("polynomial-time approximation scheme" (PTAS))

general TSP cannot be approximated within any factor $\alpha(n)$ unless $P = NP$

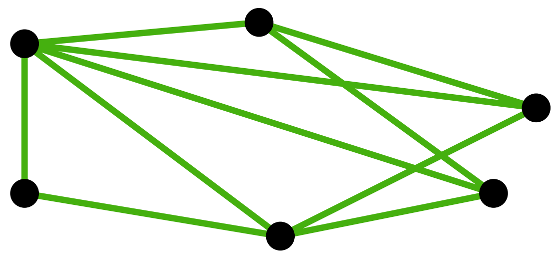
Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

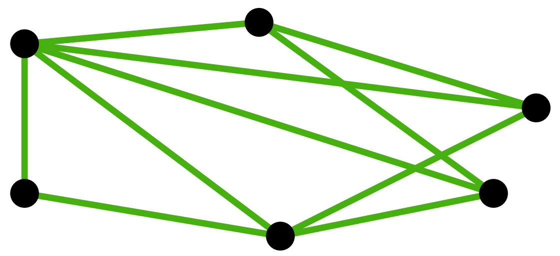
Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)

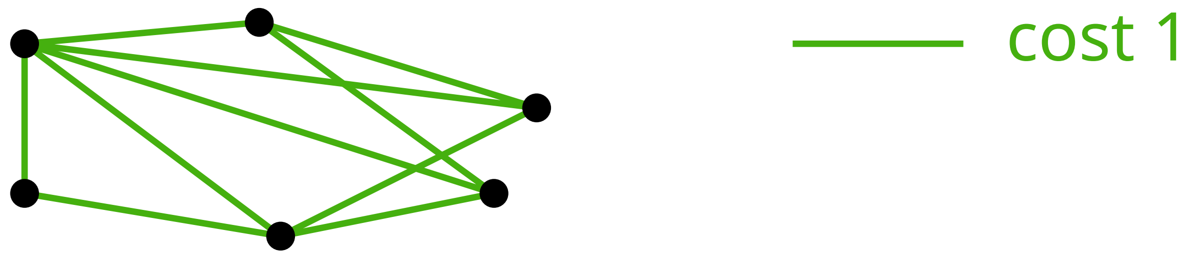


Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Non-approximability of general TSP

Assume "for the sake of contradiction": r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)

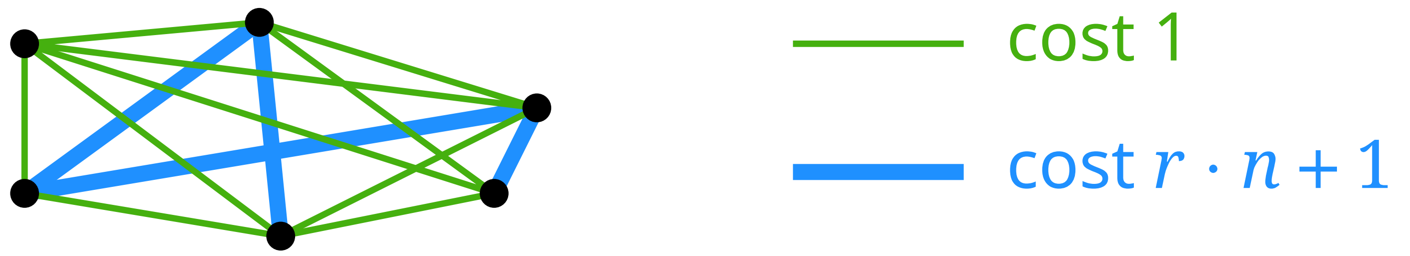


Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)

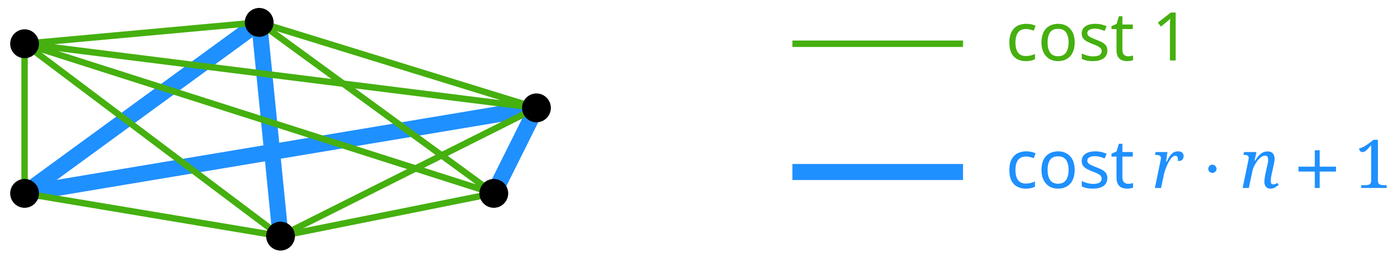


Define TSP instance $G' = (V, E')$ with edge costs $w(\{i, j\}) = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



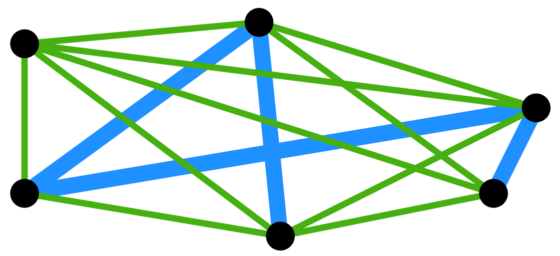
Define TSP instance $G' = (V, E')$ with edge costs $w(\{i, j\}) = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost

Non-approximability of general TSP

Assume "for the sake of contradiction": r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



cost 1

cost $r \cdot n + 1$

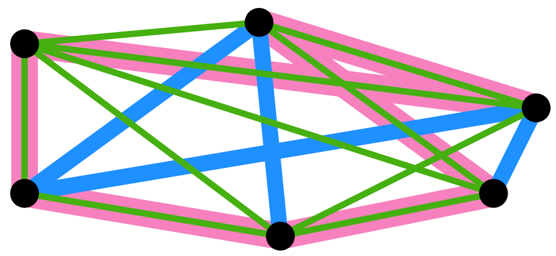
Define TSP instance $G' = (V, E')$ with edge costs $w(\{i, j\}) = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

Non-approximability of general TSP

Assume "for the sake of contradiction": r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



cost 1

cost $r \cdot n + 1$

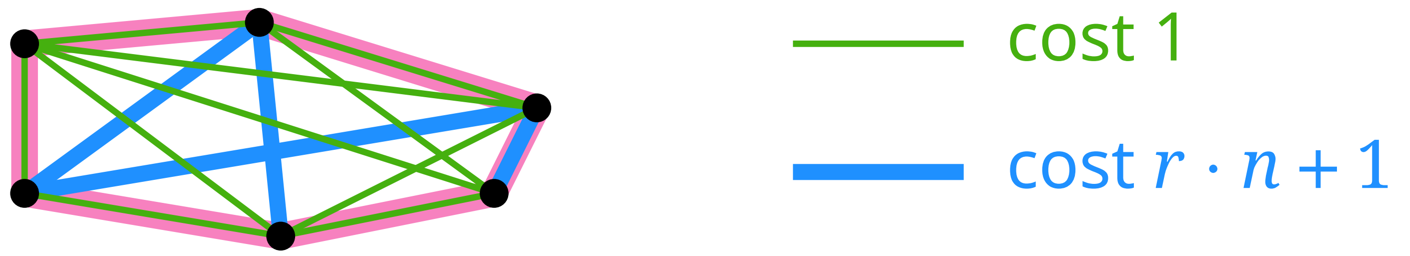
Define TSP instance $G' = (V, E')$ with edge costs $w(\{i, j\}) = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



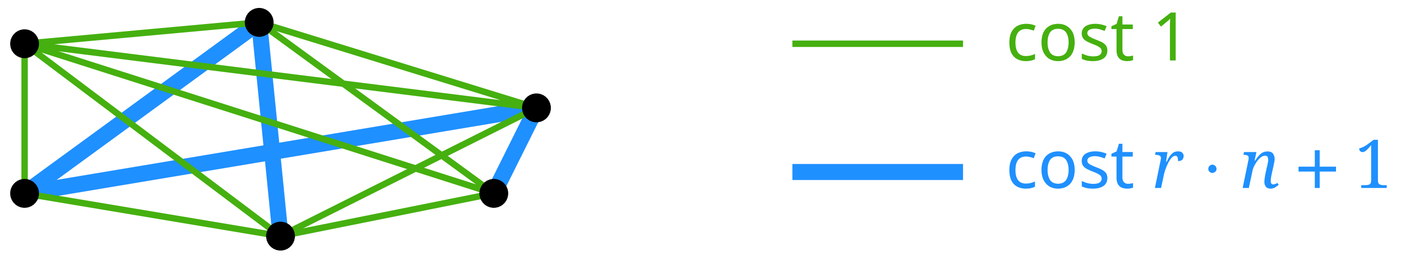
Define TSP instance $G' = (V, E')$ with edge costs $w(\{i, j\}) = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

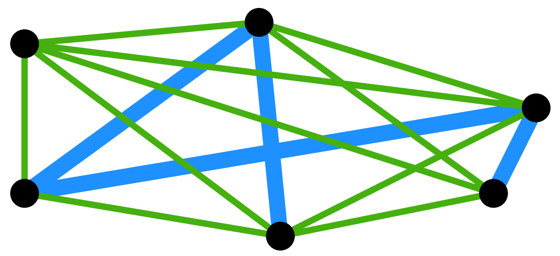
Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

$G \in \text{HC} \Leftrightarrow \text{cost}(A(G')) \leq r \cdot n$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



cost 1

cost $r \cdot n + 1$

Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

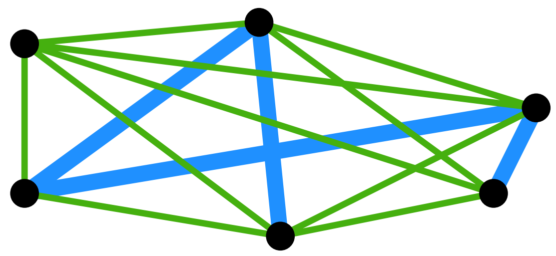
$G \in \text{HC} \Leftrightarrow \text{cost}(A(G')) \leq r \cdot n$

$G \notin \text{HC} \Leftrightarrow \text{cost}(A(G')) > r \cdot n$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



cost 1

cost $r \cdot n + 1$

Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

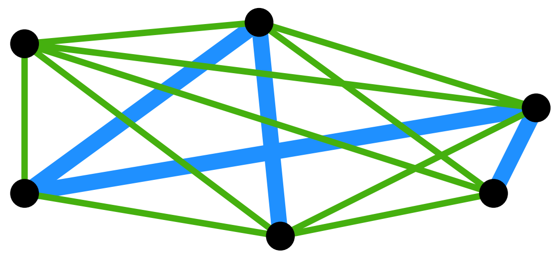
Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

$$\left.\begin{array}{l} G \in \text{HC} \Leftrightarrow \text{cost}(A(G')) \leq r \cdot n \\ G \notin \text{HC} \Leftrightarrow \text{cost}(A(G')) > r \cdot n \end{array}\right\} \Rightarrow r\text{-Approx. for TSP } \textit{implies} \text{ HC} \in P$$

Non-approximability of general TSP

Assume “for the sake of contradiction”: r-approximation algorithm A for TSP

Given $G = (V, E)$ instance of Hamiltonian Cycle Problem (HC)



cost 1

cost $r \cdot n + 1$

Define TSP instance $G' = (V, E')$ with edge costs $w(\{i,j\}) = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ r \cdot n + 1 & \text{otherwise} \end{cases}$

Observation: opt. tour for G' has cost $\begin{cases} n & \text{if } G \in \text{HC} \\ \geq (r \cdot n + 1) + (n - 1) & \text{if } G \notin \text{HC} \end{cases}$

$$\left.\begin{array}{l} G \in \text{HC} \Leftrightarrow \text{cost}(A(G')) \leq r \cdot n \\ G \notin \text{HC} \Leftrightarrow \text{cost}(A(G')) > r \cdot n \end{array}\right\} \begin{array}{l} \Rightarrow r\text{-Approx. for TSP } \textit{implies } \text{HC} \in P \\ \Rightarrow P = NP \end{array}$$

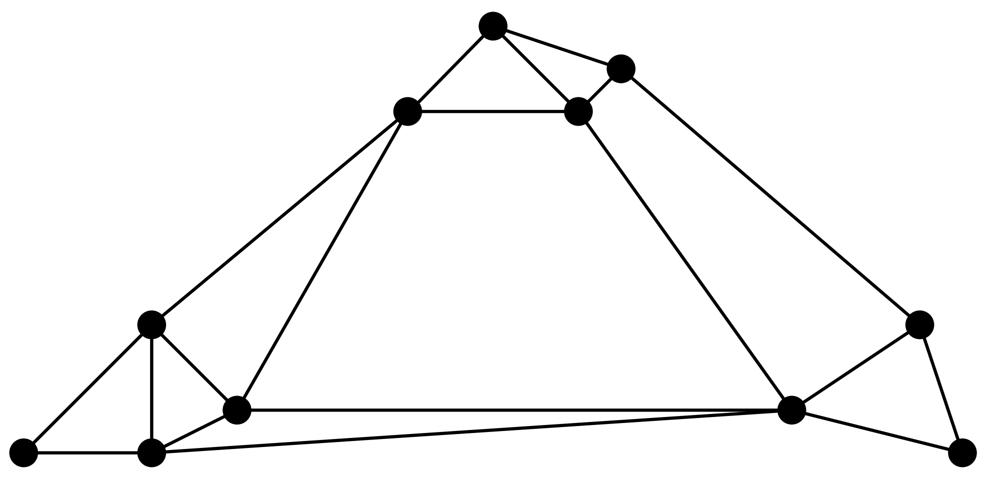
MultiwayCut

MultiwayCut

Given: A connected graph G

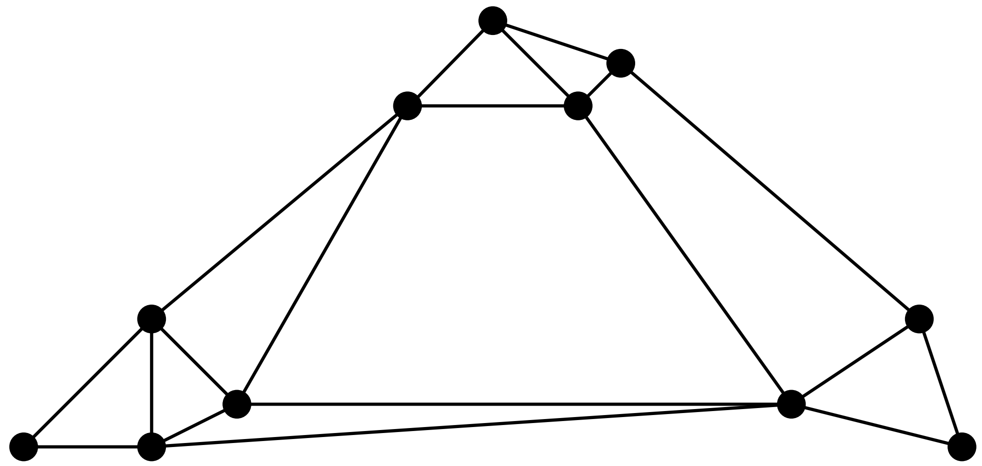
MultiwayCut

Given: A connected graph G



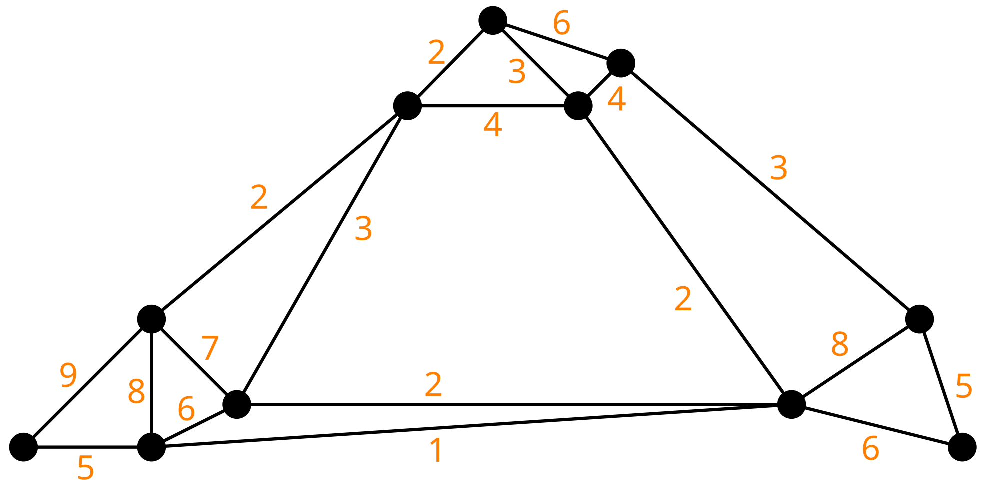
MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$



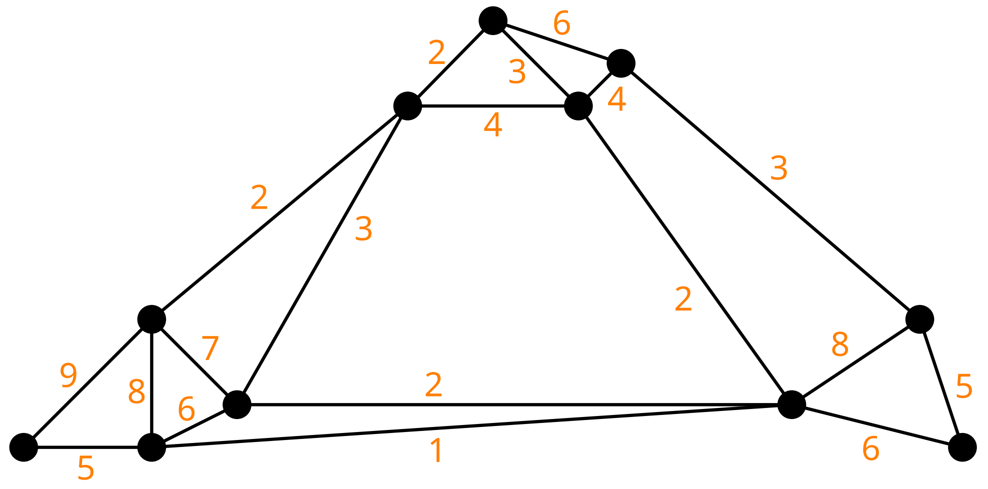
MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$



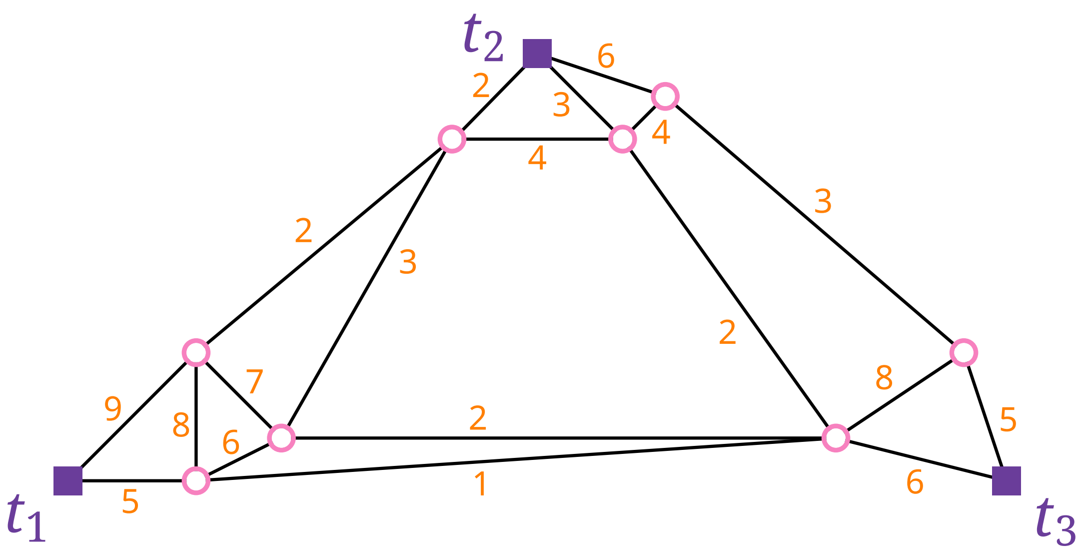
MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.



MultiwayCut

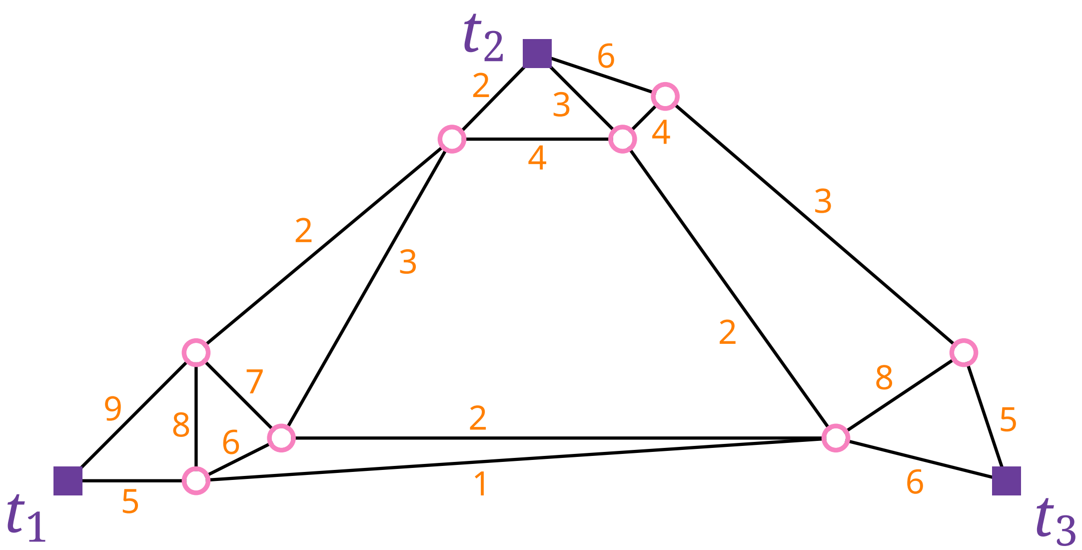
Given: A connected graph G with edge costs $c\colon E(G) \longrightarrow \mathbb{Q}^+$
and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.



MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \longrightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

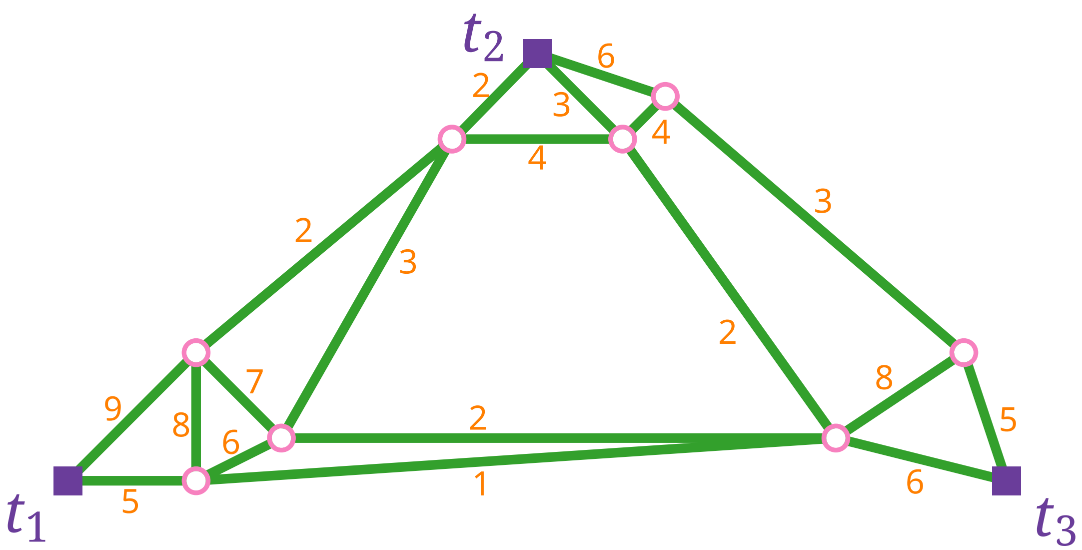
A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.



MULTIWAYCUT

Given: A connected graph G with edge costs $c\colon E(G) \longrightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

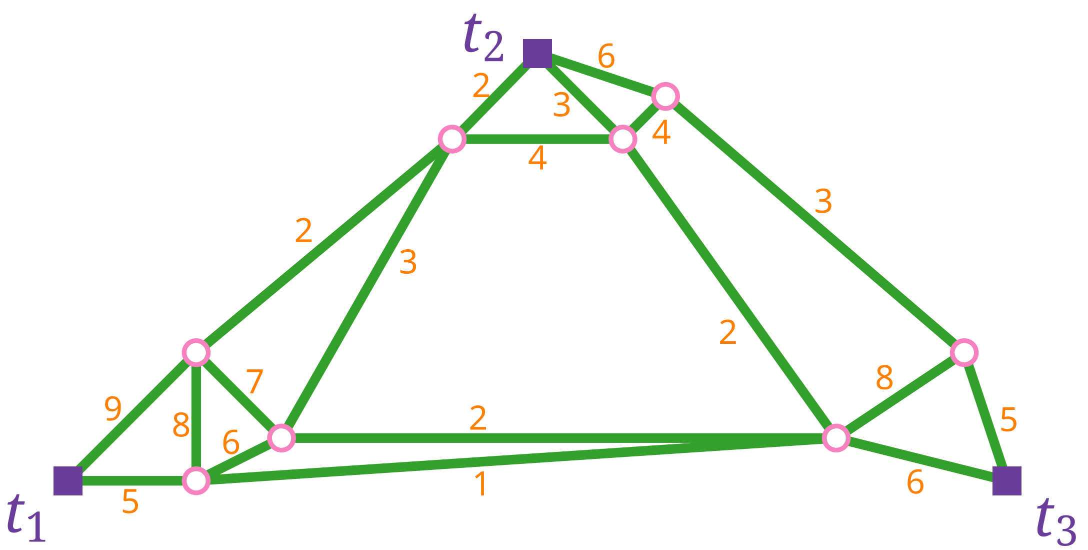


MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.

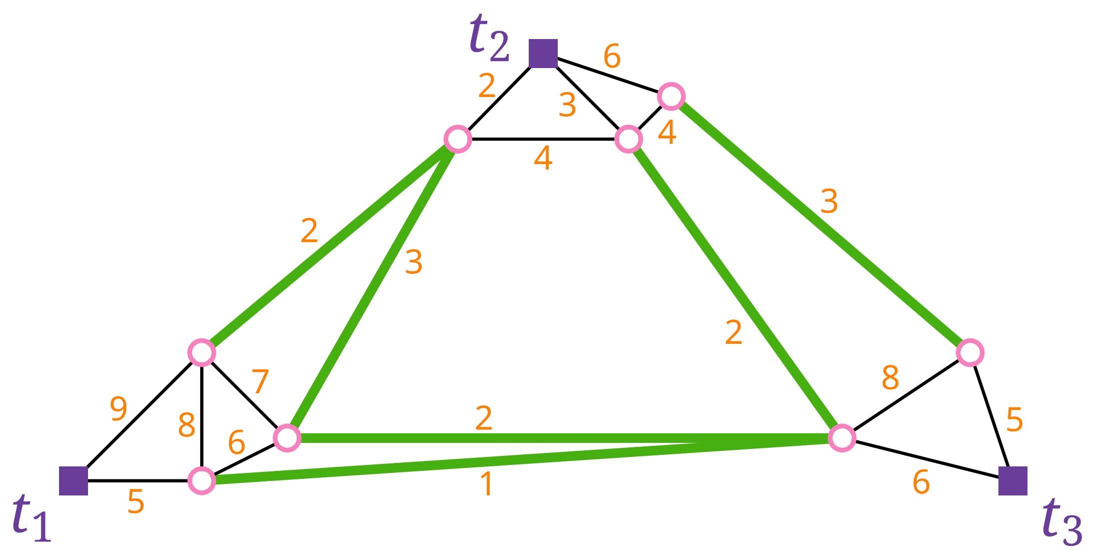


MULTIWAYCUT

Given: A connected graph G with edge costs $c\colon E(G) \longrightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.

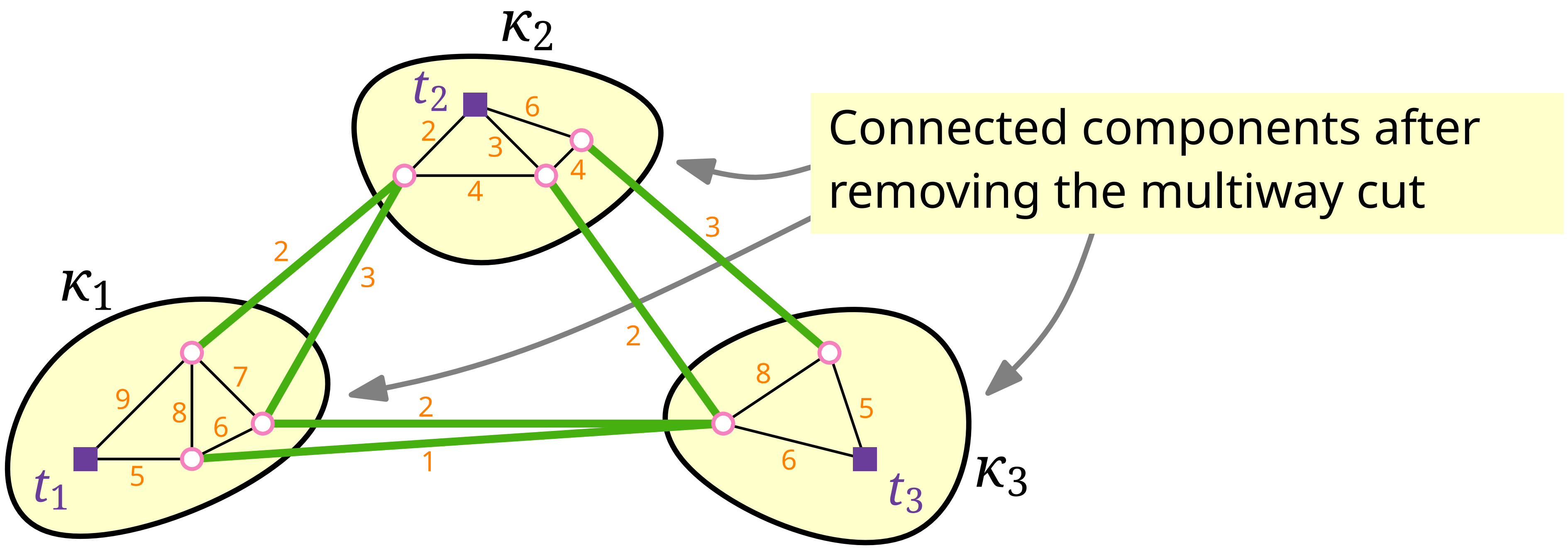


MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.

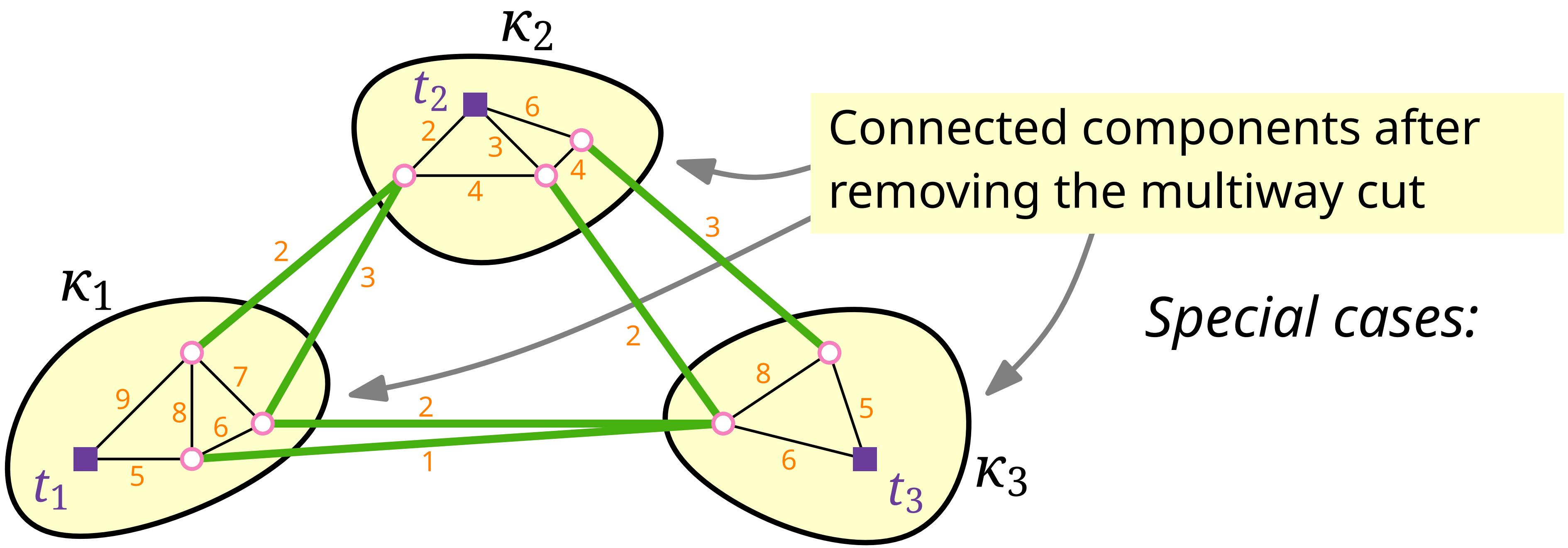


MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.



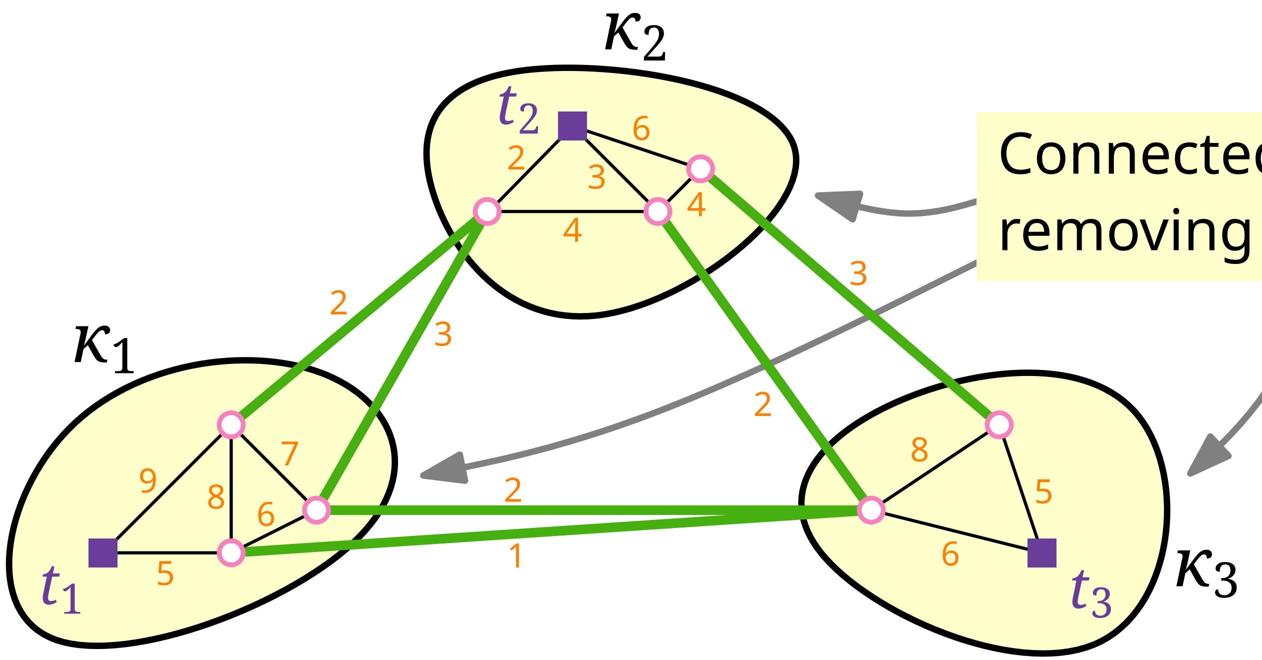
Special cases:

MultiwayCut

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.



Connected components after removing the multiway cut

Special cases:

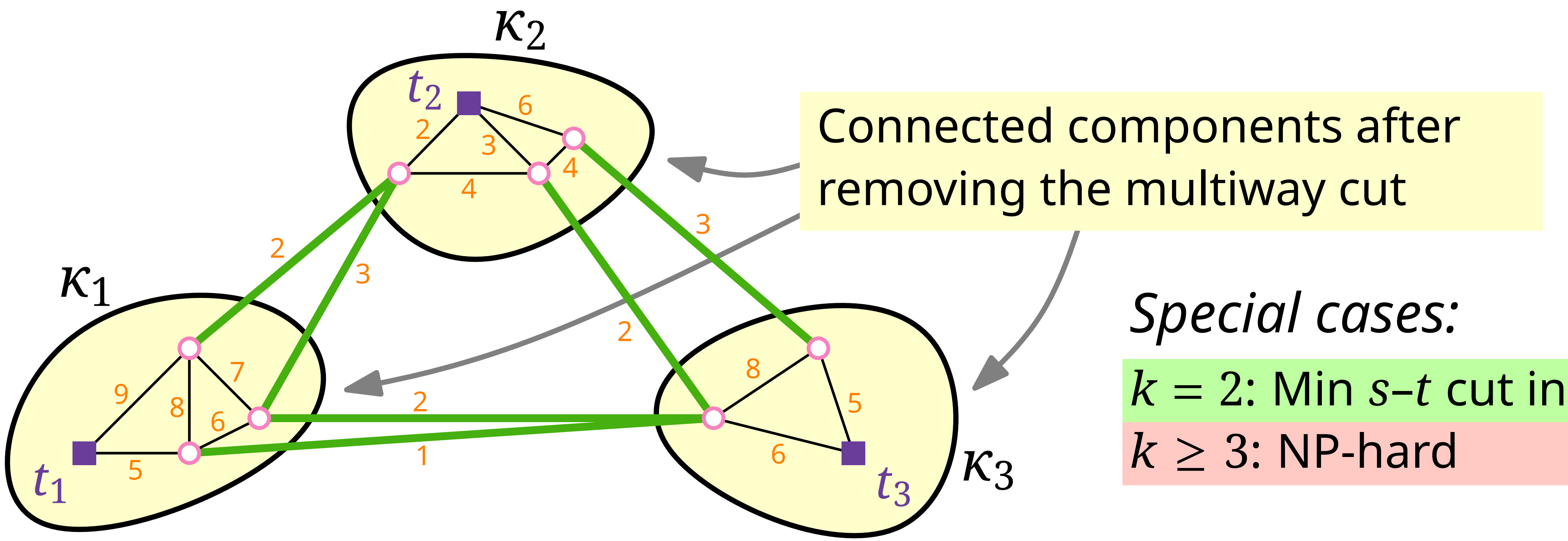
$k = 2$: Min s–t cut in P

MULTIWAYCUT

Given: A connected graph G with edge costs $c\colon E(G) \to \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A multiway cut of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A minimum-cost multiway cut of T.



Special cases:

$k = 2$: Min s–t cut in P

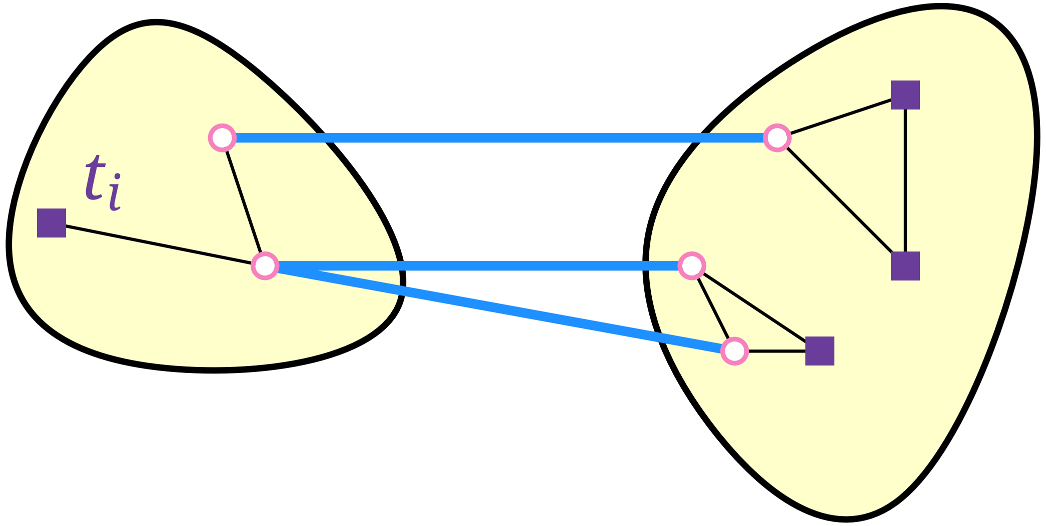
$k \geq 3$: NP-hard

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

Isolating Cuts

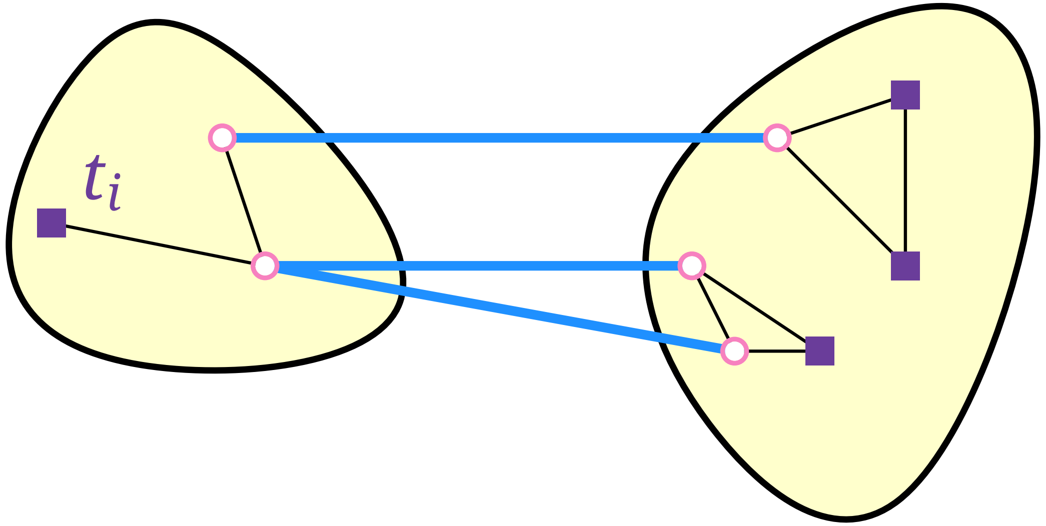
An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.



Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges
that disconnects t_i from all other terminals.

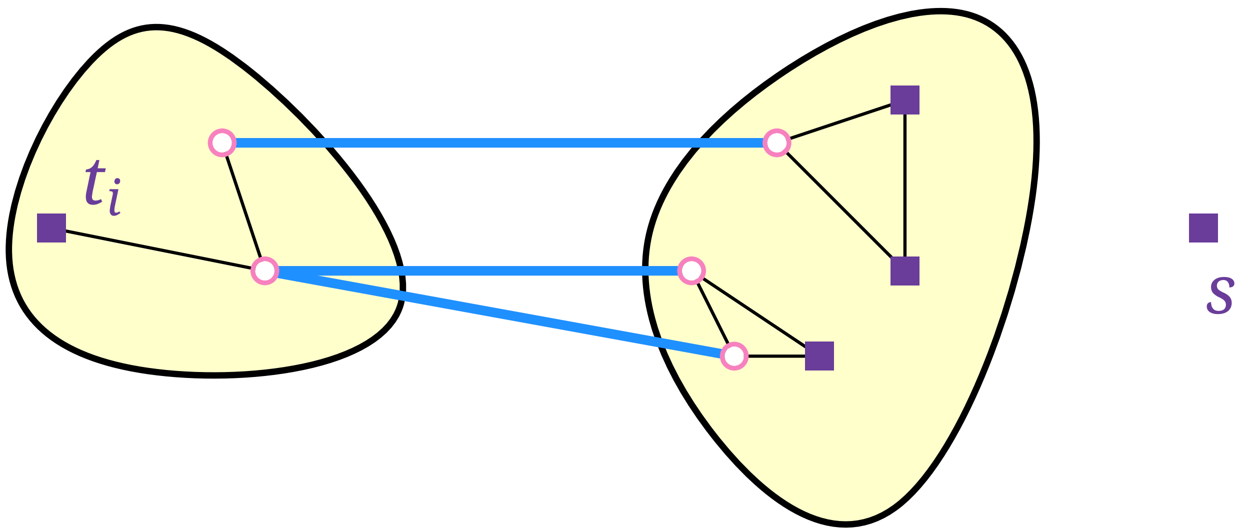
A minimum-cost isolating cut for t_i can be computed efficiently:



Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges
that disconnects t_i from all other terminals.

A minimum-cost isolating cut for t_i can be computed efficiently:

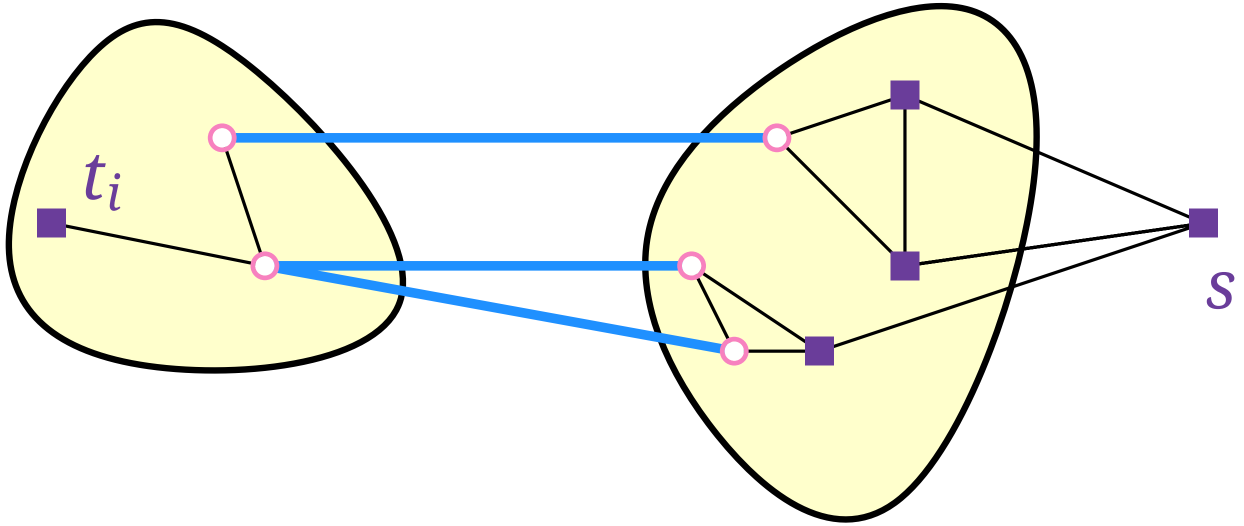


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost isolating cut for t_i can be computed efficiently:

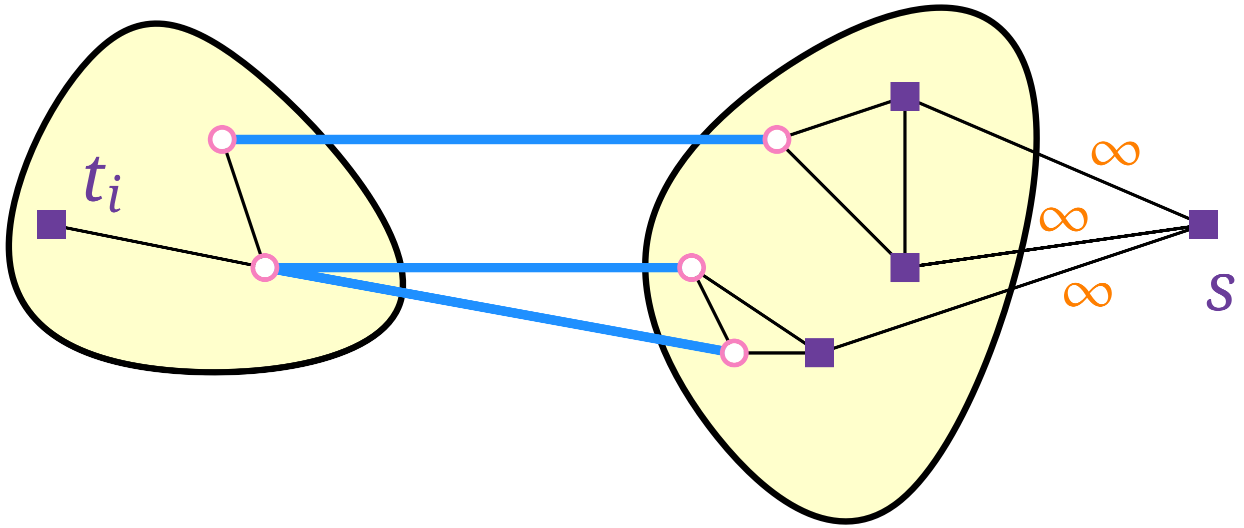


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges
that disconnects t_i from all other terminals.

A minimum-cost isolating cut for t_i can be computed efficiently:

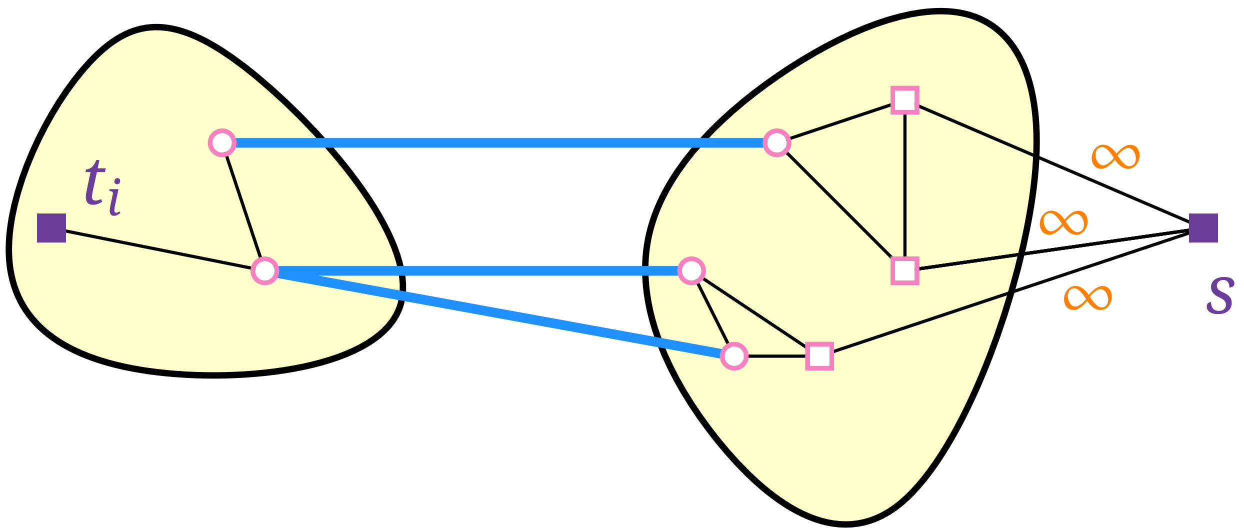


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost isolating cut for t_i can be computed efficiently:

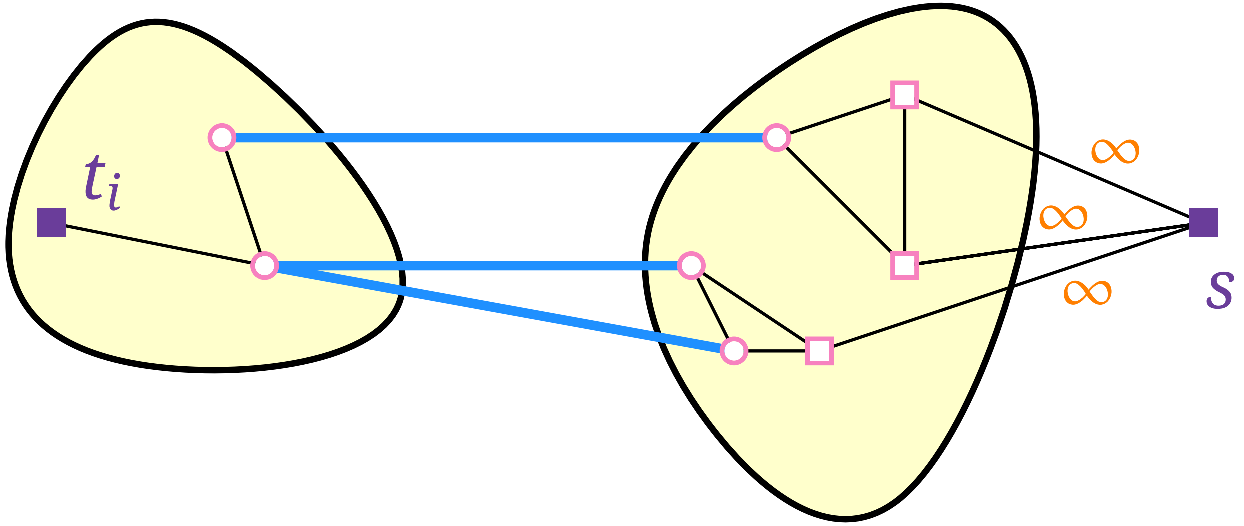


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost isolating cut for t_i can be computed efficiently:



Add dummy terminal s and find a minimum-cost s–t_i cut.

Algorithm for MULTIWAYCUT

Algorithm MultiwayCut

For $i = 1, \ldots, k$:

Algorithm MULTIWAYCUT

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.

Algorithm MultiwayCut

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k-1$ cheapest such isolating cuts.

Algorithm MultiwayCut

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k - 1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

Algorithm MULTIWAYCUT

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k-1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

$$\Rightarrow c(C) \quad ? \quad \sum_{i=1}^{k} c(C_i)$$

Algorithm MULTIWAYCUT

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k-1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

$$\Rightarrow c(C) \leq \sum_{i=1}^{k} c(C_i)$$

Algorithm MULTIWAYCUT

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k-1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

$\Rightarrow c(C) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} c(C_i)$ because:

Algorithm MultiwayCut

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k - 1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

$\Rightarrow c(C) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} c(C_i)$ because:

for the most expensive cut of $C_1, \ldots, C_k$, say C_1, we have

$$c(C_1) \geq$$

Algorithm MULTIWAYCUT

For $i = 1, \ldots, k$:

- Compute a minimum-cost isolating cut C_i for t_i.
- Return the union C of the $k - 1$ cheapest such isolating cuts.

In other words:
Ignore the most expensive one of the isolating cuts $C_1, \ldots, C_k$.

$$\Rightarrow c(C) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} c(C_i) \qquad \text{because:}$$

for the most expensive cut of $C_1, \ldots, C_k$, say C_1, we have

$$c(C_1) \geq \frac{1}{k} \sum_{i=1}^{k} c(C_i).$$

Approximation Factor

Theorem. This algorithm is a factor-approximation algorithm for MULTIWAYCUT.

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Approximation Factor

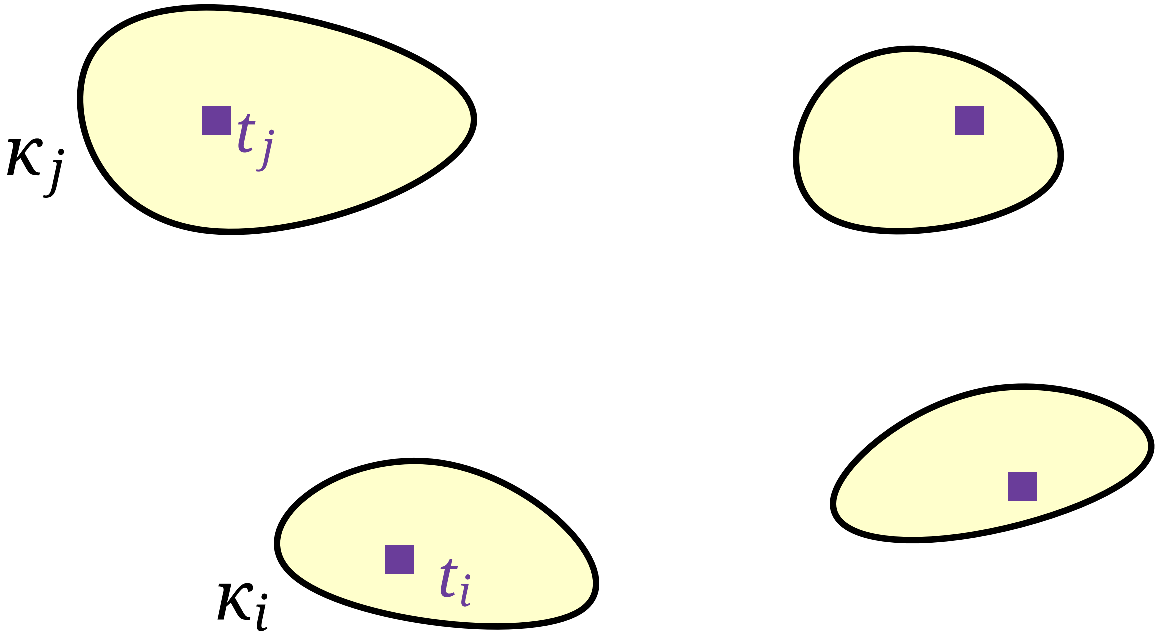
Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

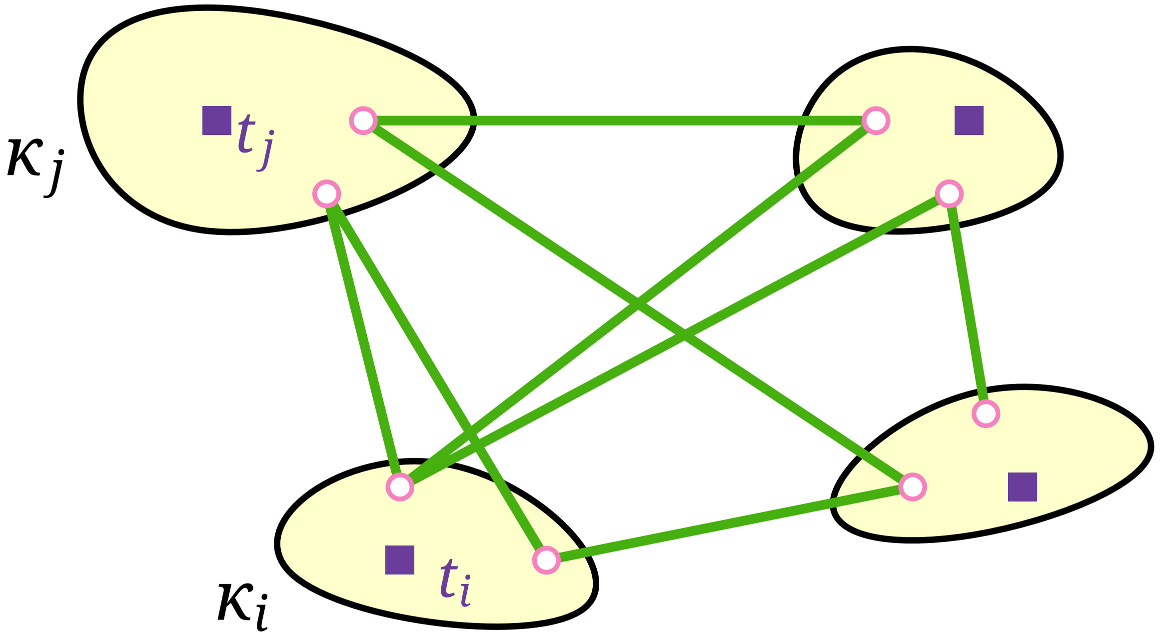
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

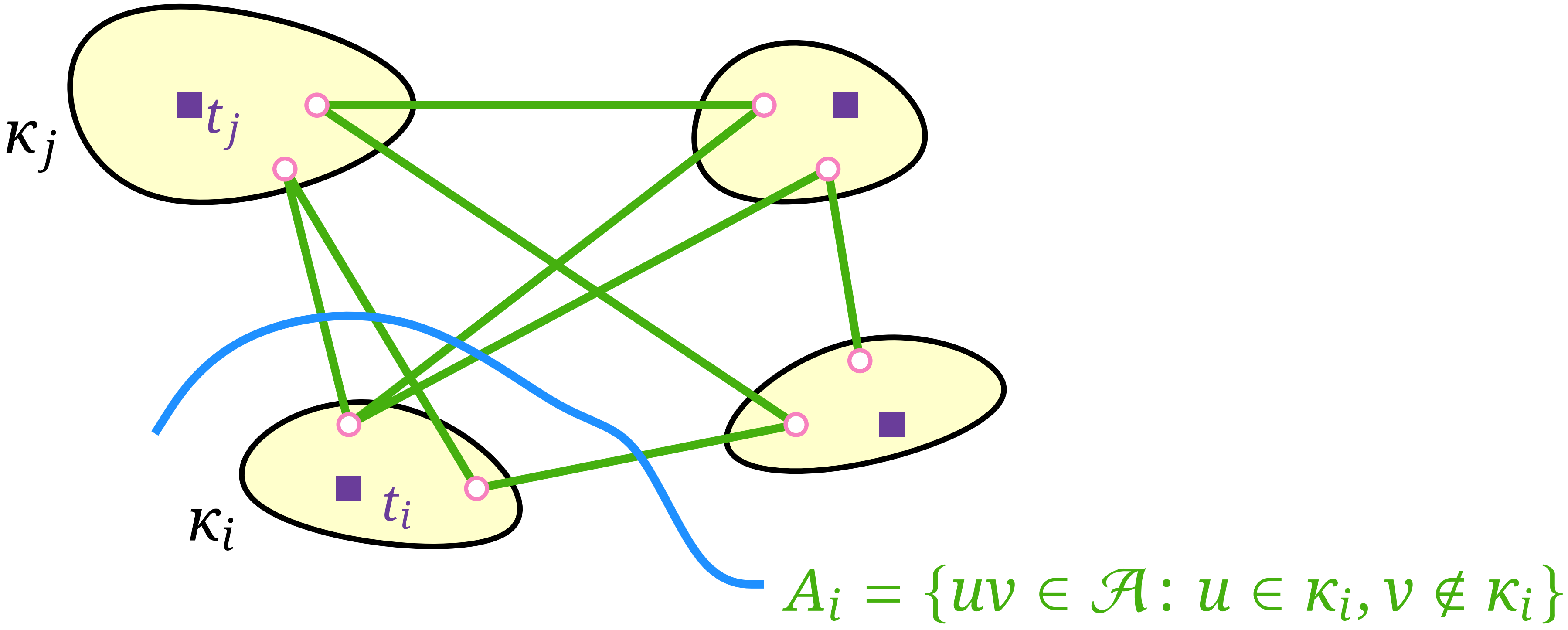
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

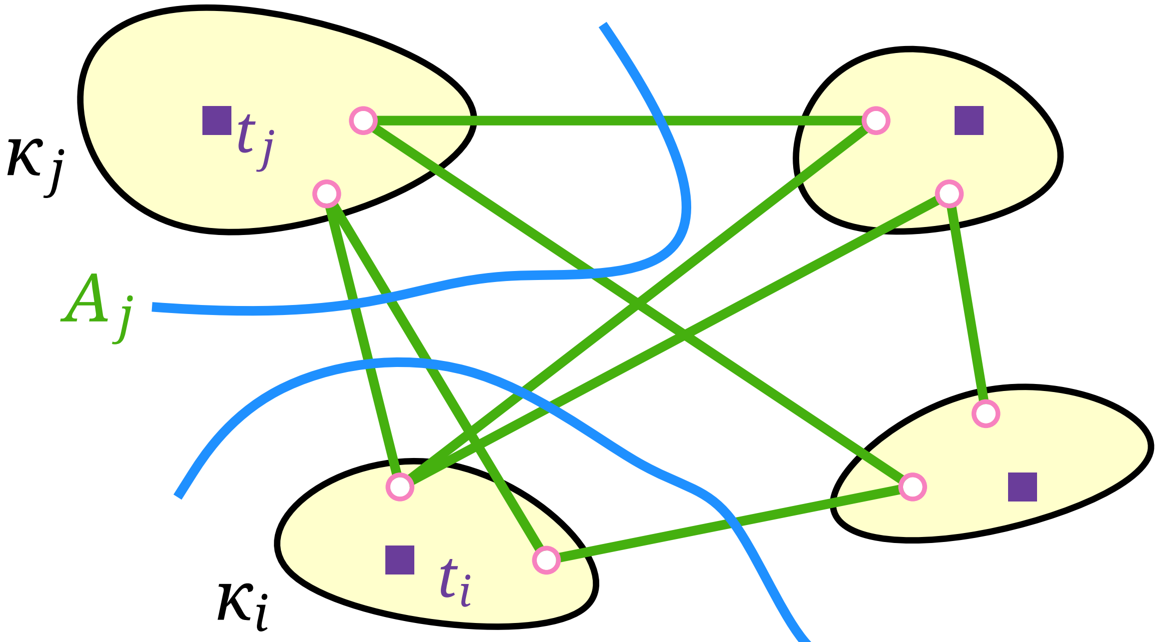
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

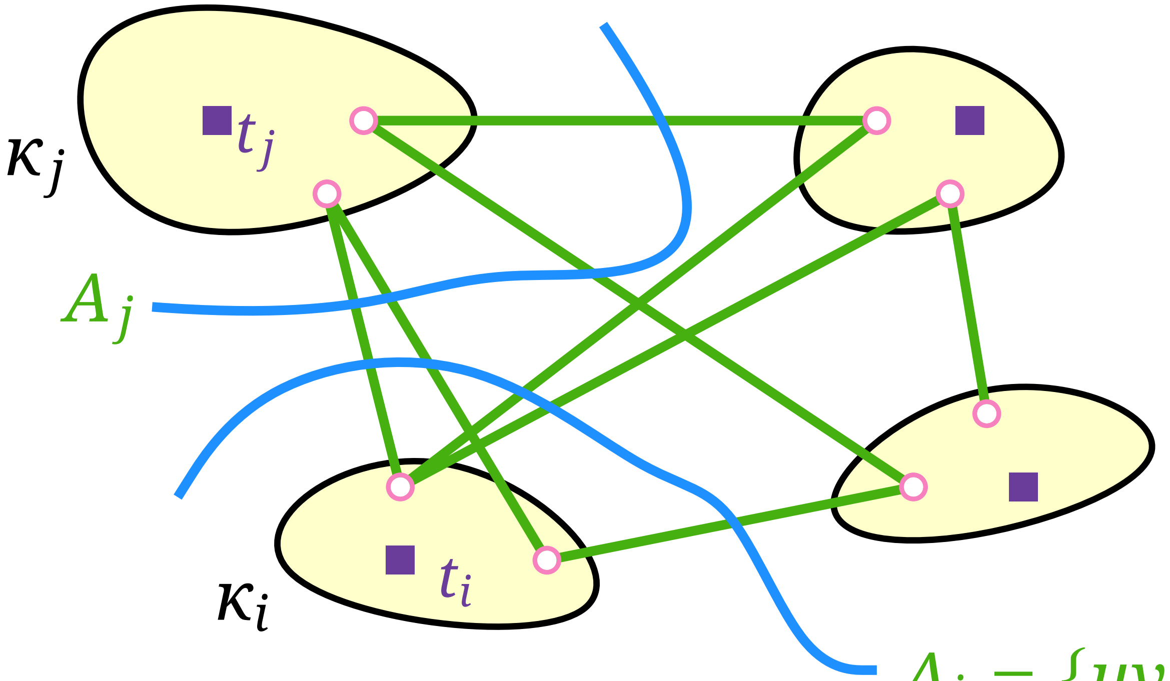


$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:



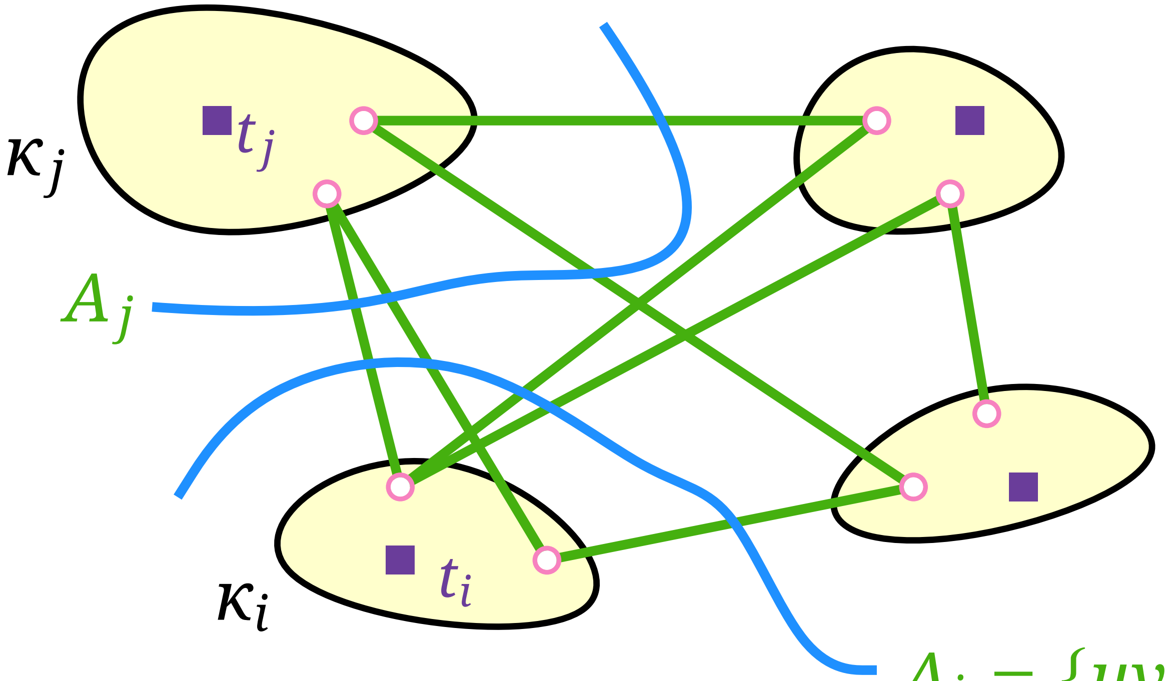
$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$

Observation. $\mathcal{A} =$

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:



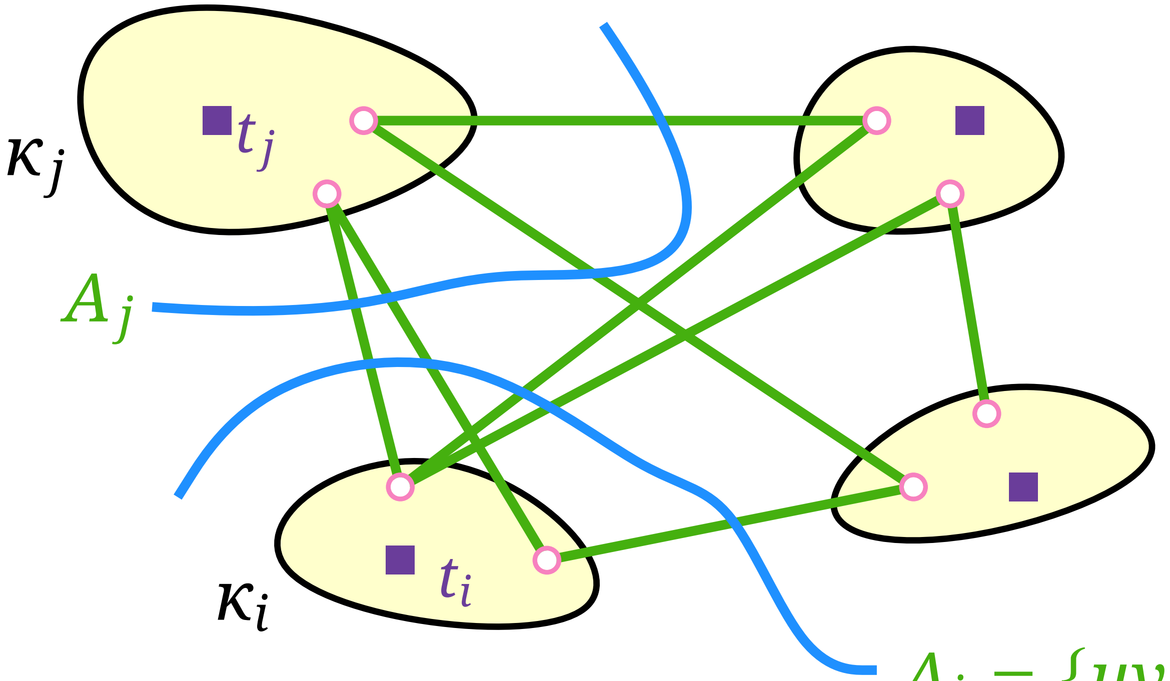
$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$

Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$

Approximation Factor

Theorem. This algorithm is a factor-$(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:



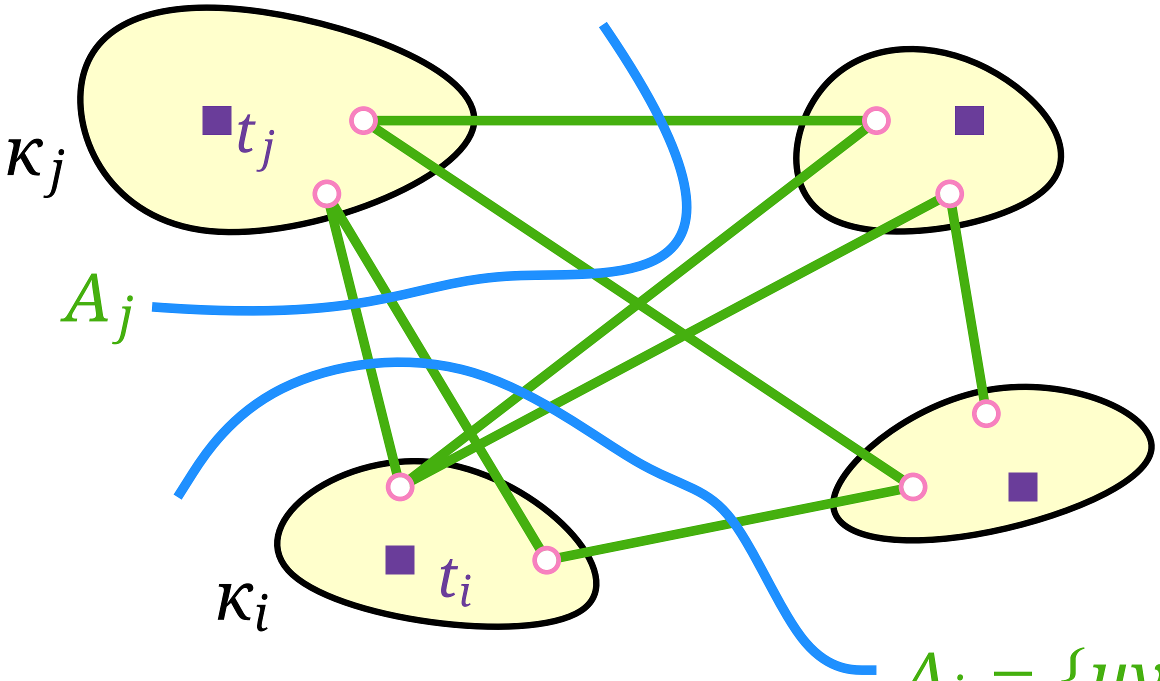
$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$

Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^{k} c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution C:



$c(C) \leq$

$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$

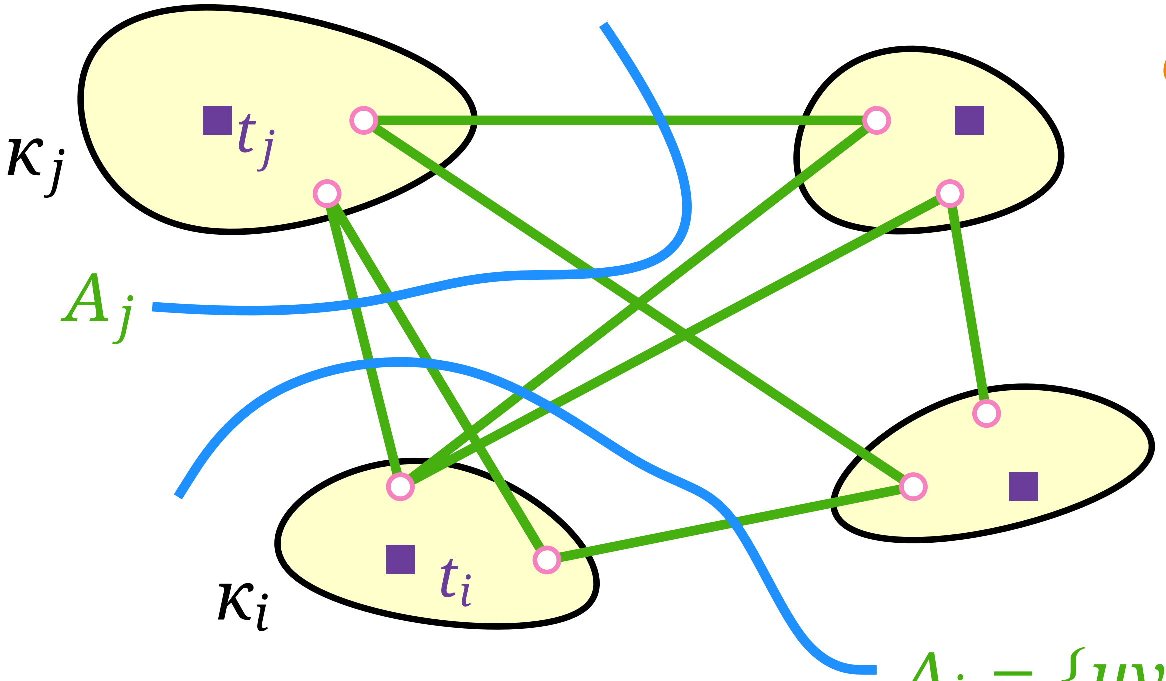
Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^{k} c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}$.

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution C:

$$c(C) \leq \left(1 - \tfrac{1}{k}\right) \sum_{i=1}^{k} c(C_i)$$



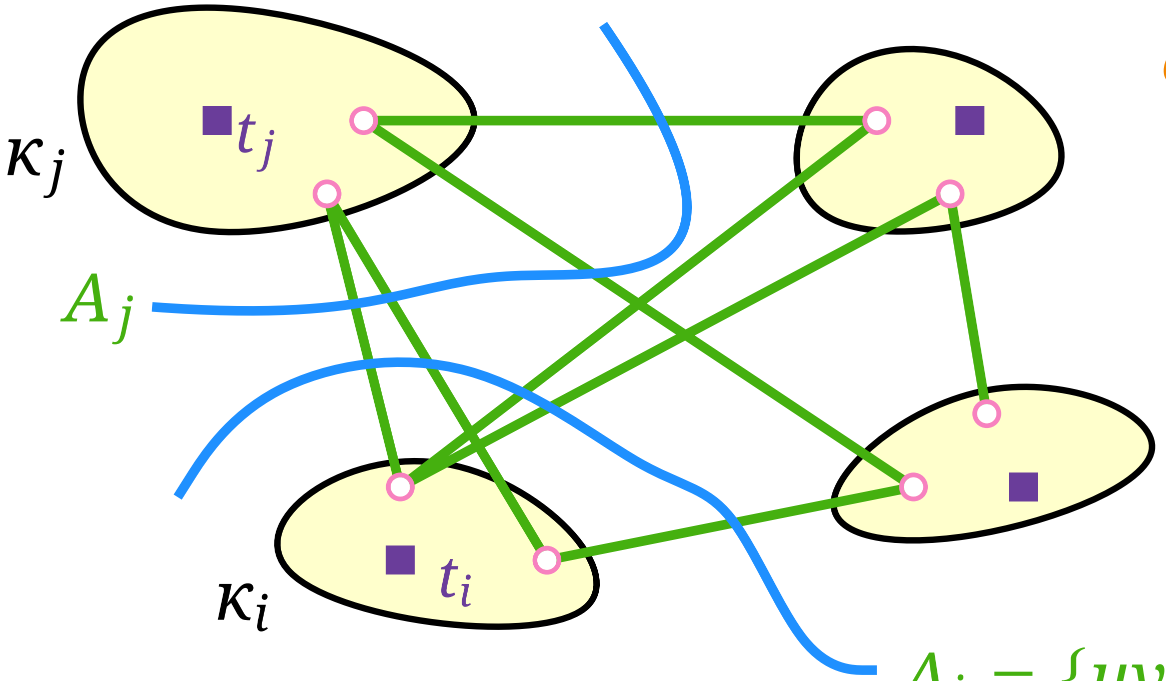
$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^{k} c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor-$(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution C:



$$c(C) \leq \left(1 - \tfrac{1}{k}\right) \textstyle\sum_{i=1}^{k} c(C_i)$$

$$\leq \left(1 - \tfrac{1}{k}\right) \textstyle\sum_{i=1}^{k} c(A_i)$$

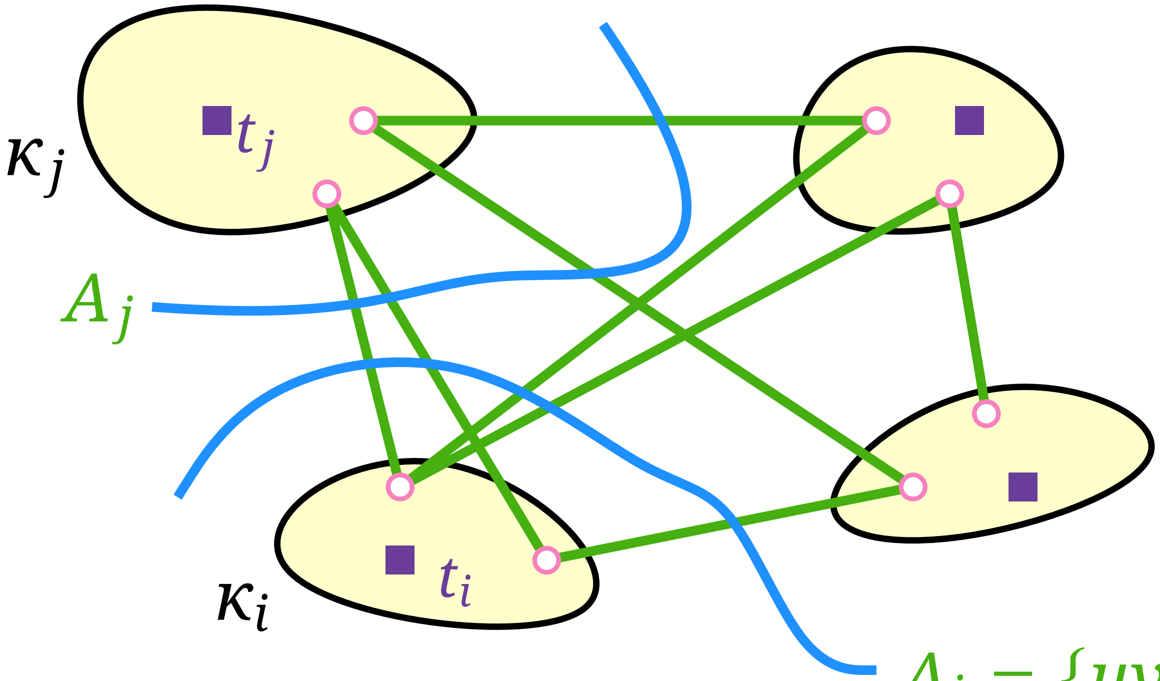
$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^{k} c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution C:



$$c(C) \leq \left(1 - \tfrac{1}{k}\right) \sum_{i=1}^{k} c(C_i)$$

$$\leq \left(1 - \tfrac{1}{k}\right) \sum_{i=1}^{k} c(A_i)$$

$$= \left(1 - \tfrac{1}{k}\right) \cdot 2 \cdot c(\mathcal{A})$$

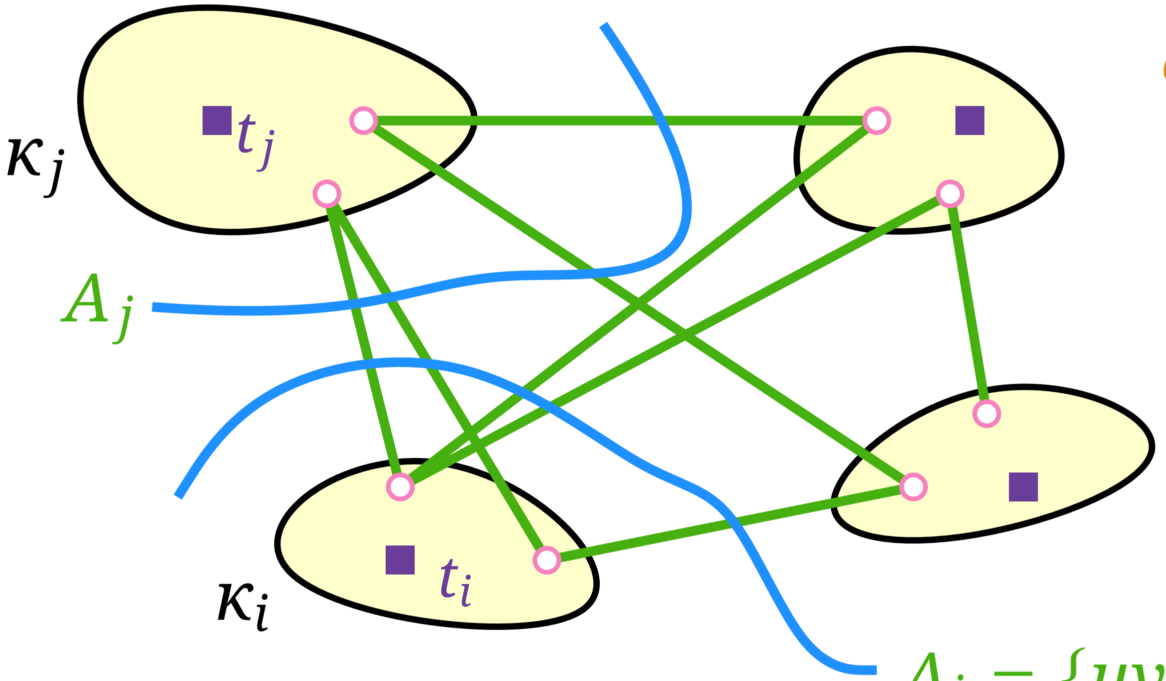
$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^{k} c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor-($2 - 2/k$) approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution C:



$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

$$\begin{aligned} c(C) &\leq \left(1 - \tfrac{1}{k}\right) \textstyle\sum_{i=1}^k c(C_i) \\ &\leq \left(1 - \tfrac{1}{k}\right) \textstyle\sum_{i=1}^k c(A_i) \\ &= \left(1 - \tfrac{1}{k}\right) \cdot 2 \cdot c(\mathcal{A}) \\ &= \left(2 - \tfrac{2}{k}\right) \cdot \mathrm{OPT} \end{aligned}$$

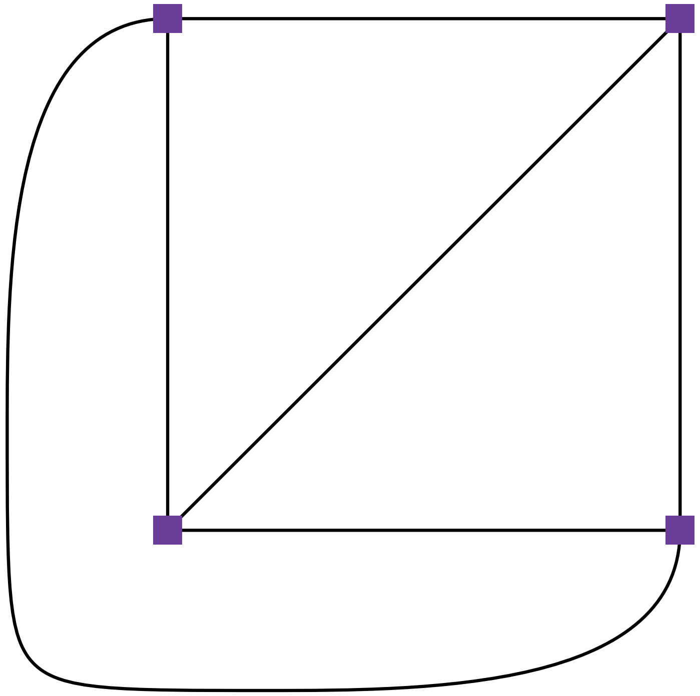
Observation. $\mathcal{A} = \bigcup_{i=1}^{k} A_i$ and $\sum_{i=1}^k c(A_i) = 2 \cdot c(\mathcal{A}) = 2 \cdot \mathrm{OPT}$.

Analysis Tight?

K_k

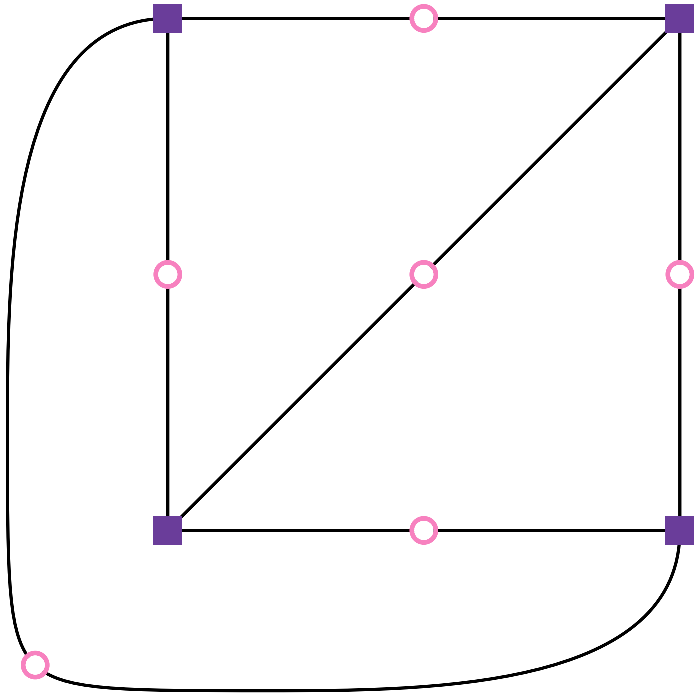
Analysis Tight?

K_k



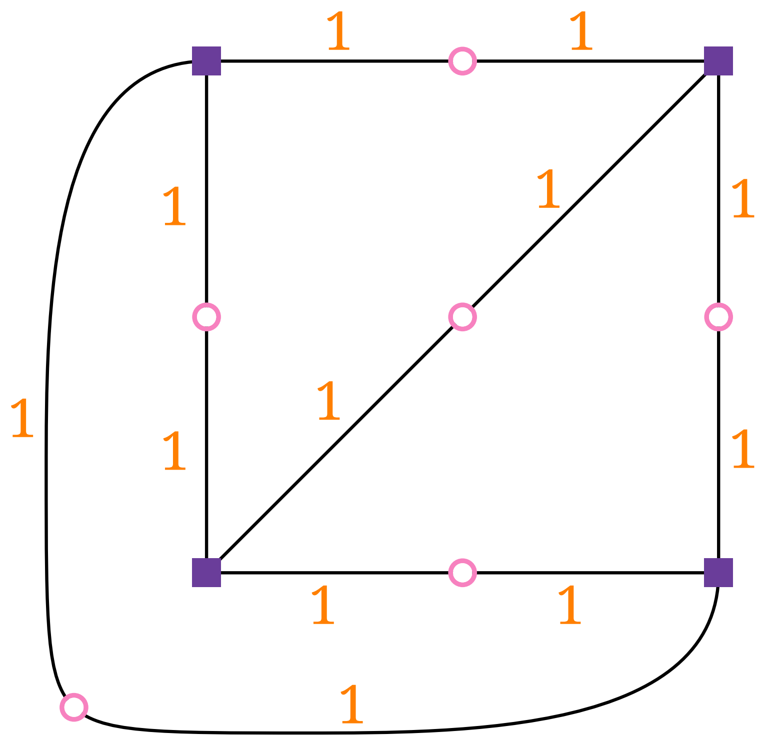
Analysis Tight?

K_k



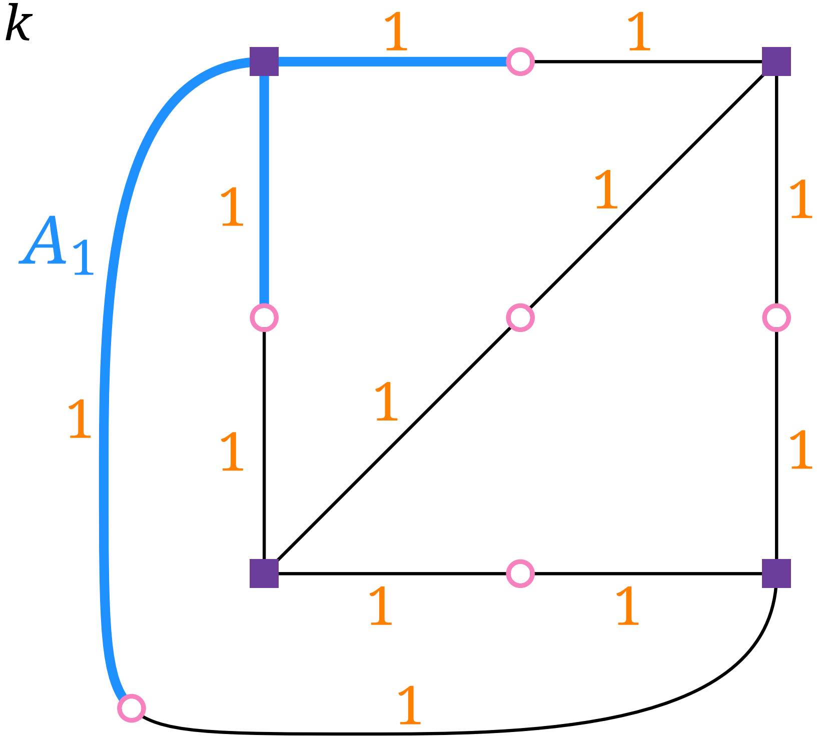
Analysis Tight?

K_k



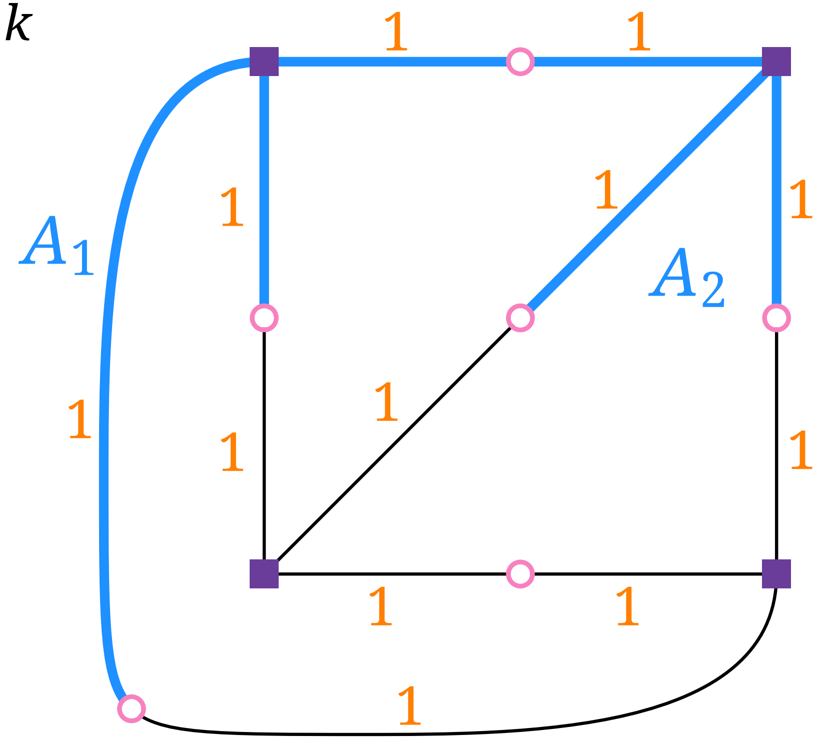
Analysis Tight?

K_k

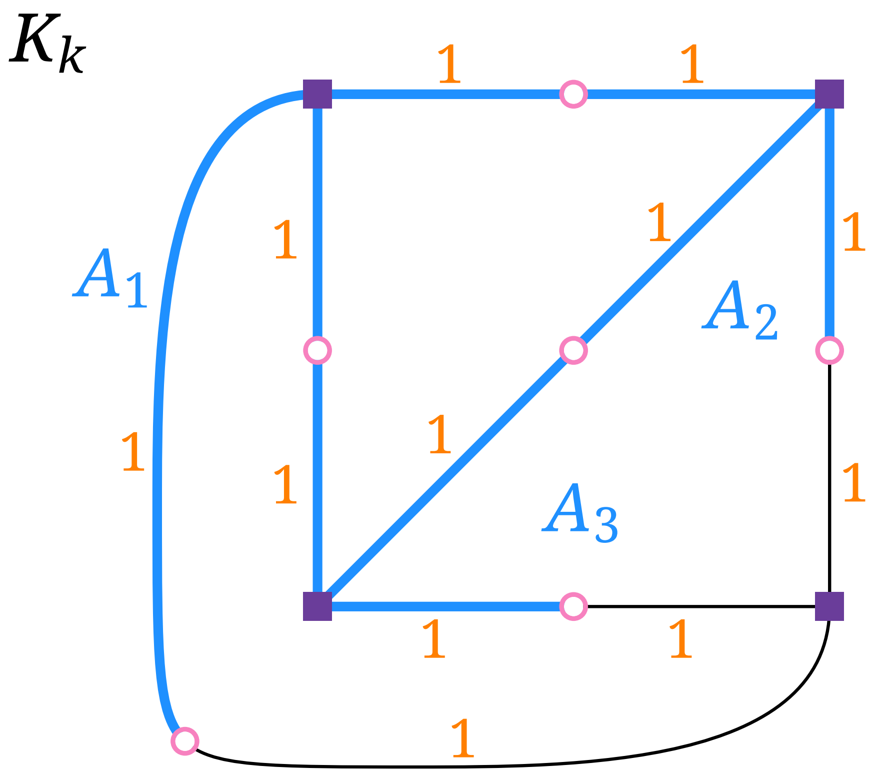


Analysis Tight?

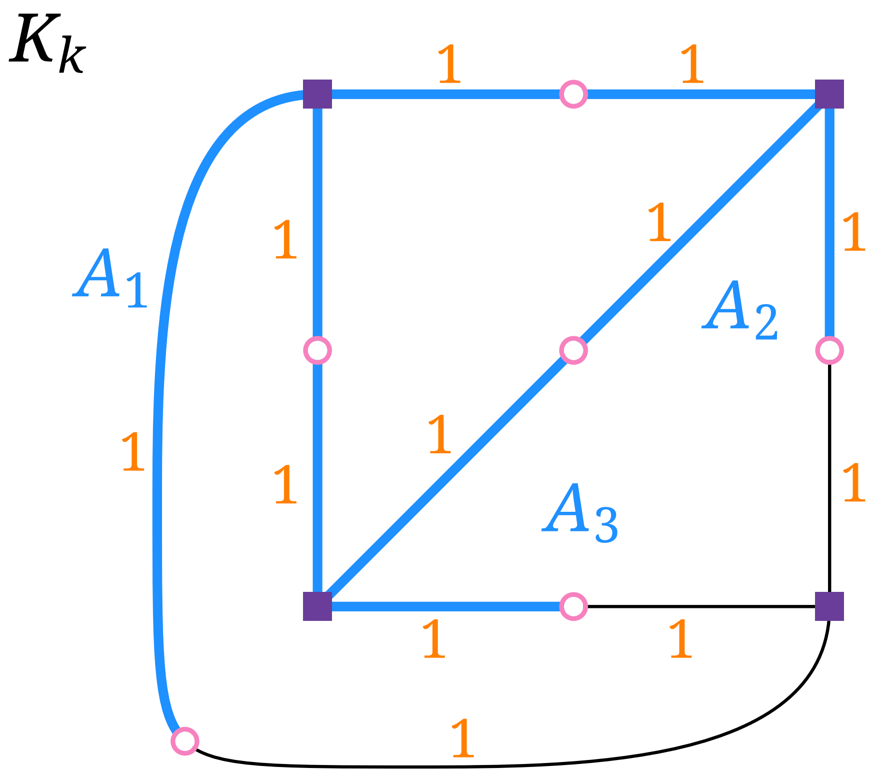
K_k



Analysis Tight?

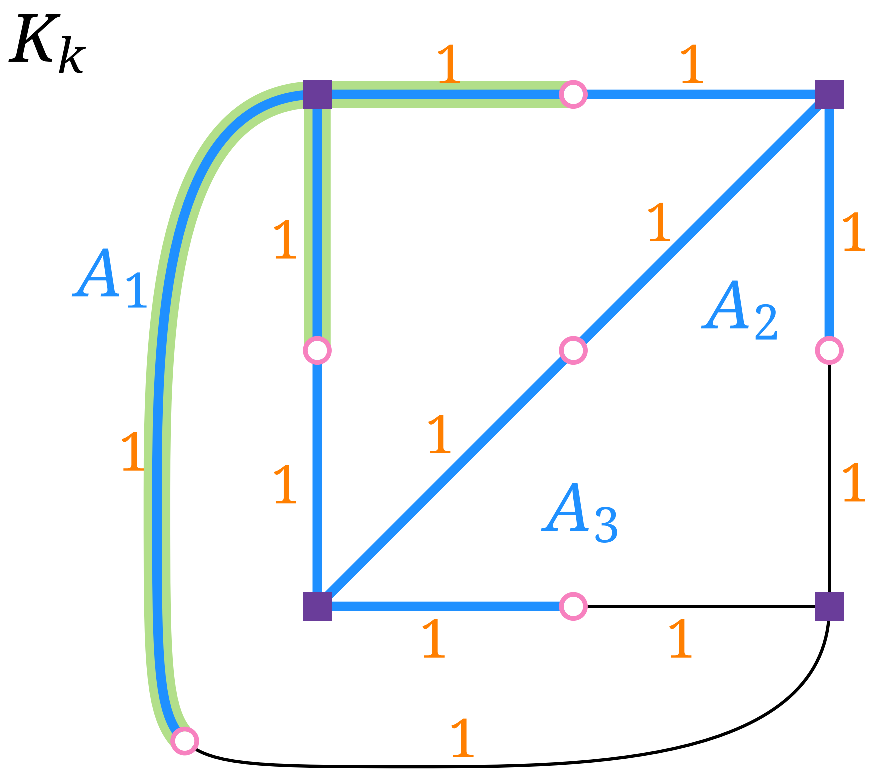


Analysis Tight?



$\text{ALG} = (k-1)(k-1)$

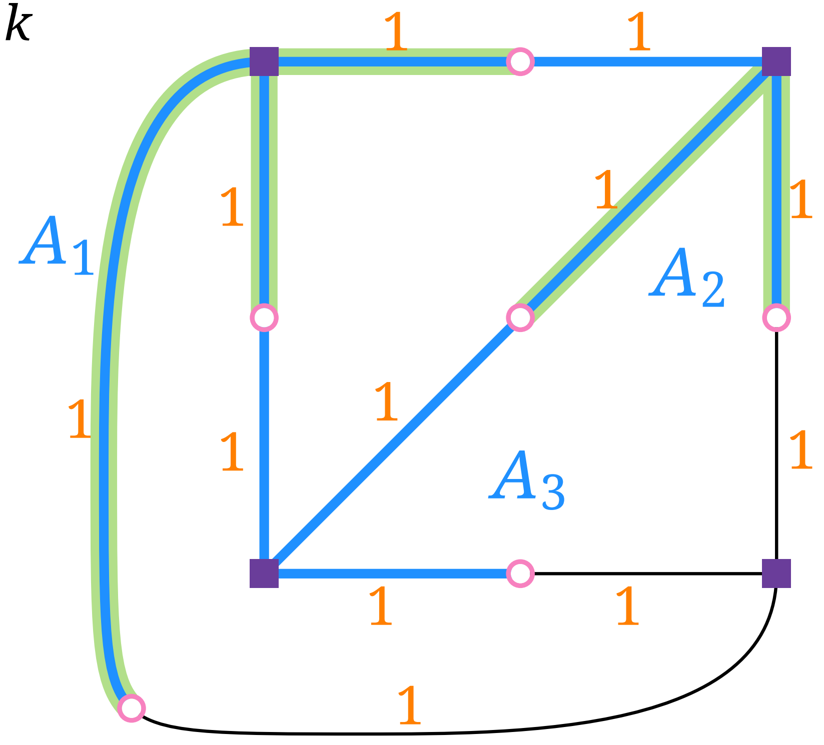
Analysis Tight?



$\mathrm{ALG} = (k-1)(k-1)$

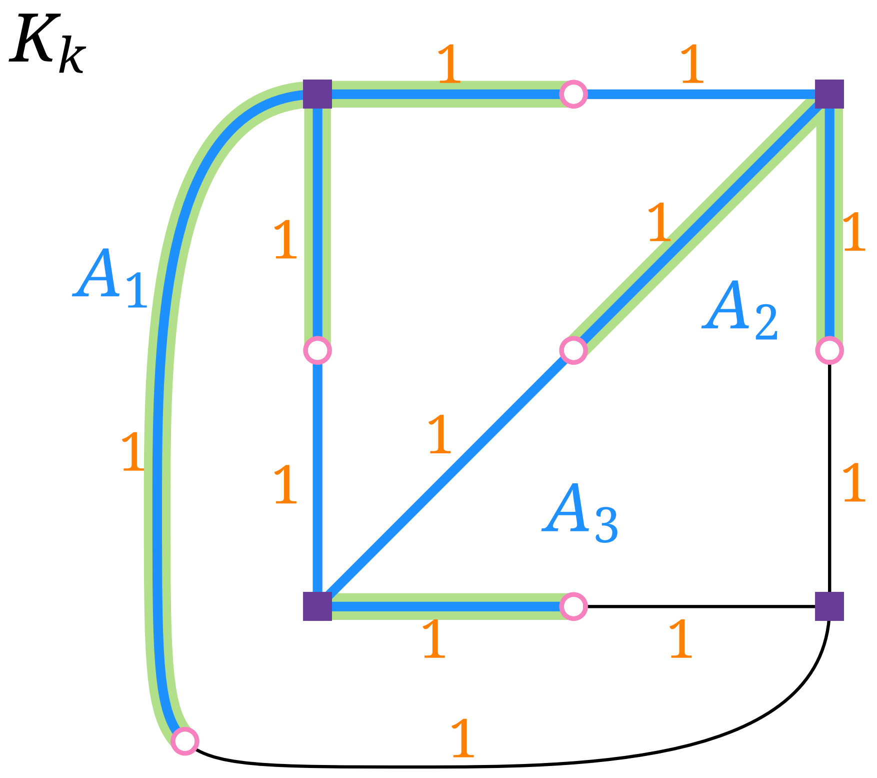
Analysis Tight?

K_k



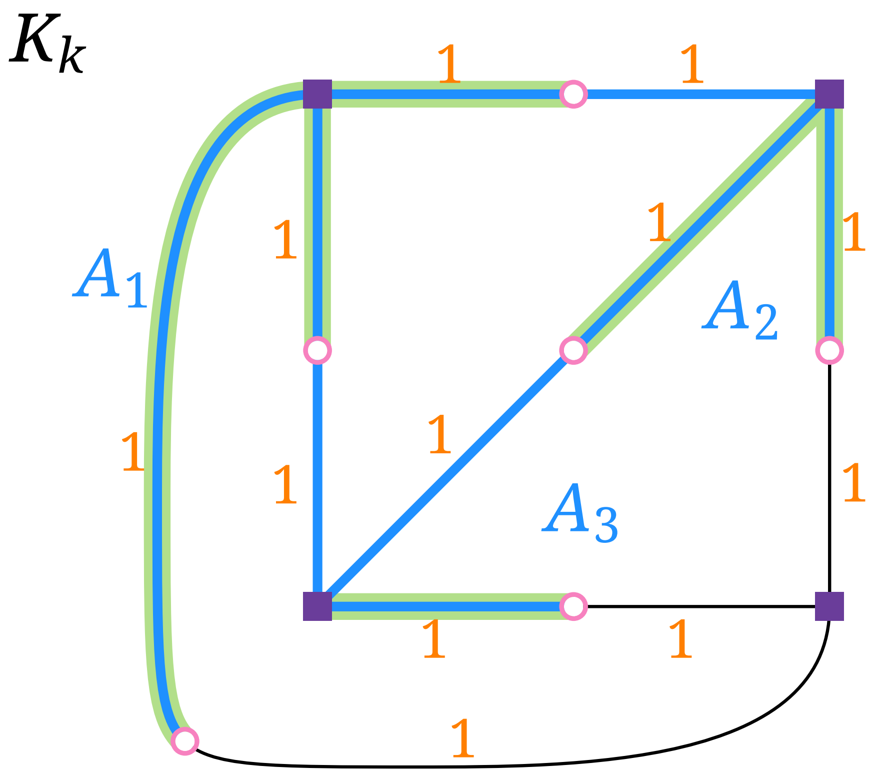
$\mathrm{ALG} = (k-1)(k-1)$

Analysis Tight?



$\text{ALG} = (k-1)(k-1)$

Analysis Tight?

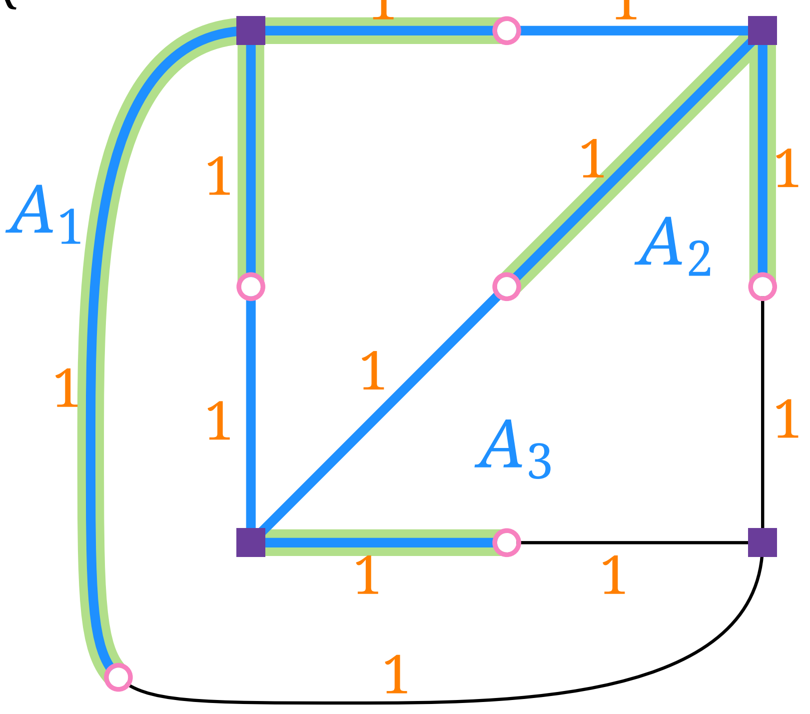


$\mathrm{ALG} = (k-1)(k-1)$

$\mathrm{OPT} = \sum_{i=1}^{k-1} i =$

Analysis Tight?

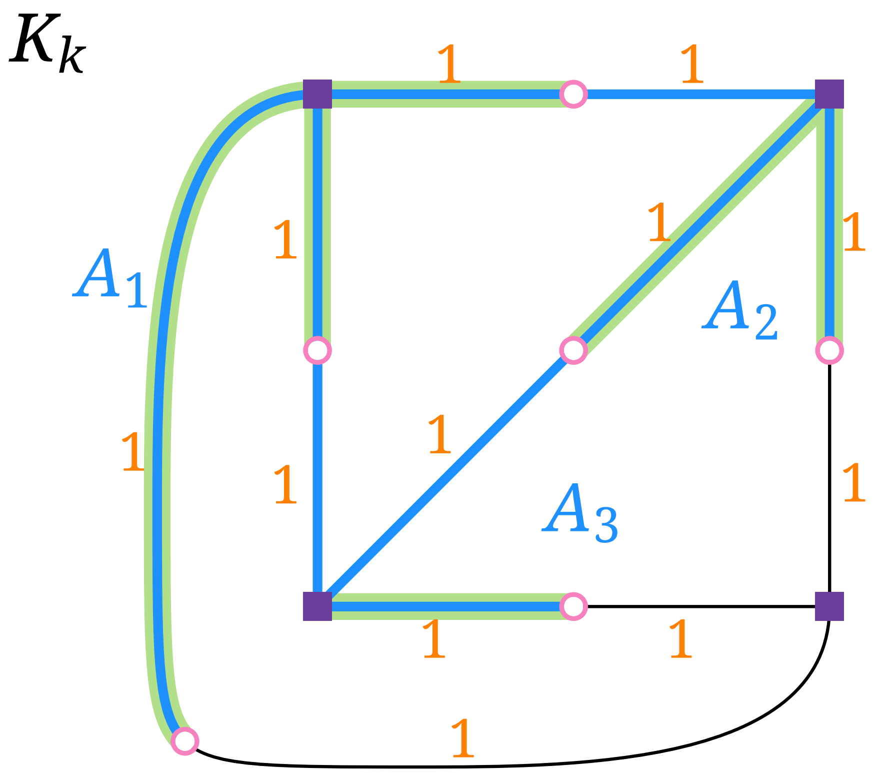
K_k



$\text{ALG} = (k-1)(k-1)$

$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$

Analysis Tight?

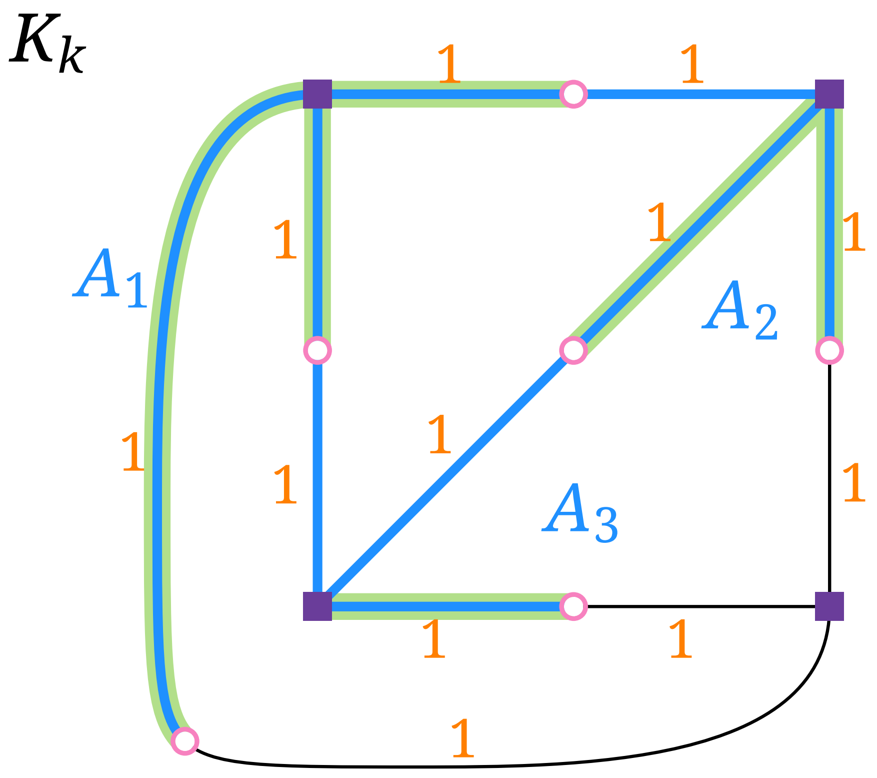


$\text{ALG} = (k-1)(k-1)$

$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k\cdot(k-1)}{2}$

$\text{ALG}/\text{OPT} =$

Analysis Tight?

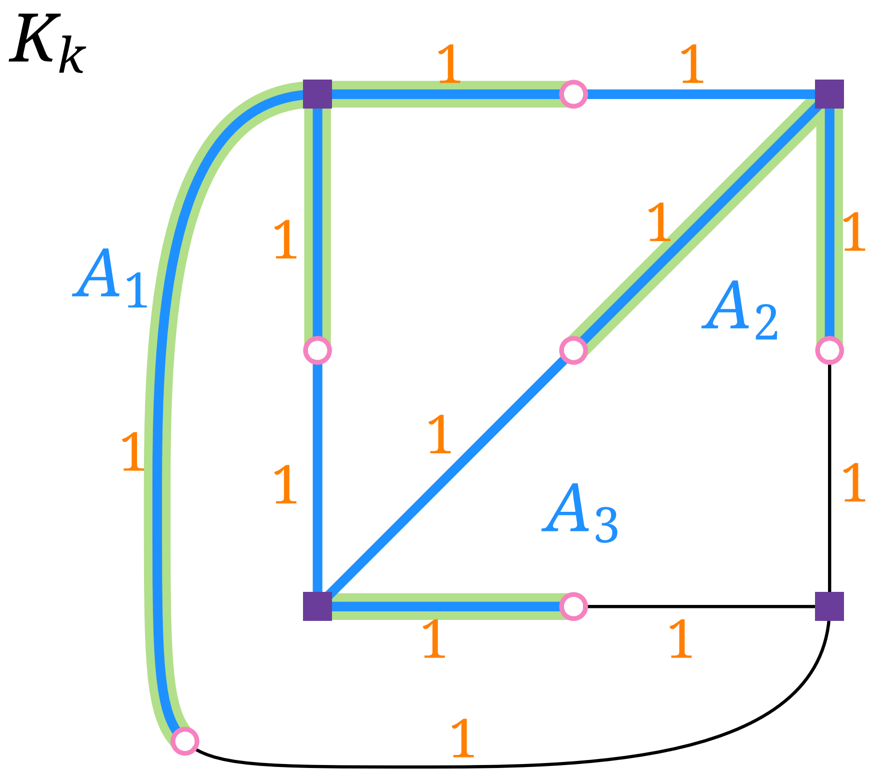


$\mathrm{ALG} = (k-1)(k-1)$

$\mathrm{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$

$\mathrm{ALG}/\mathrm{OPT} = \frac{2(k-1)}{k} =$

Analysis Tight?



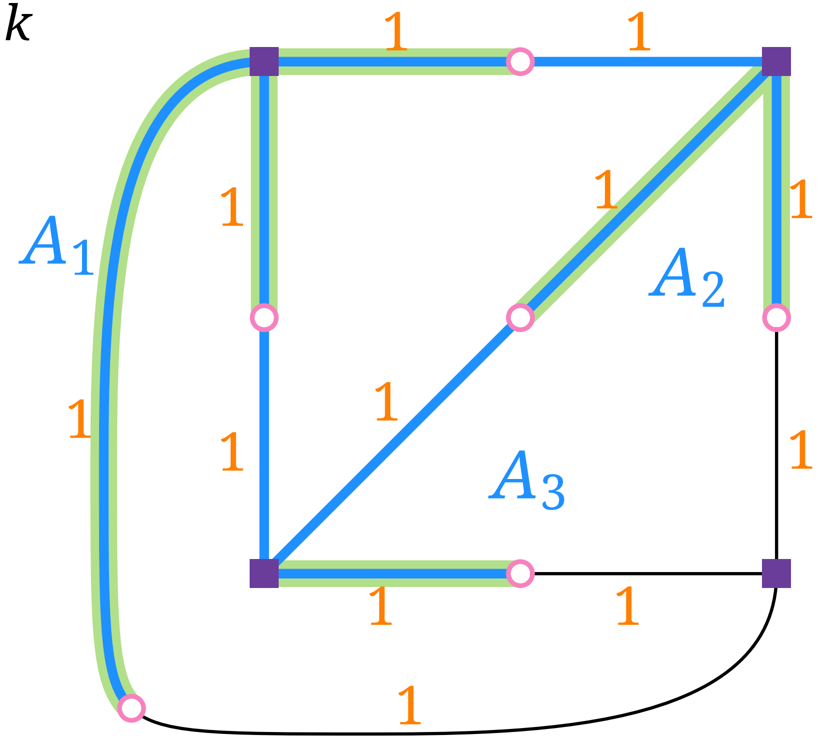
$\text{ALG} = (k-1)(k-1)$

$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k\cdot(k-1)}{2}$

$\text{ALG}/\text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$

Analysis Tight?

K_k



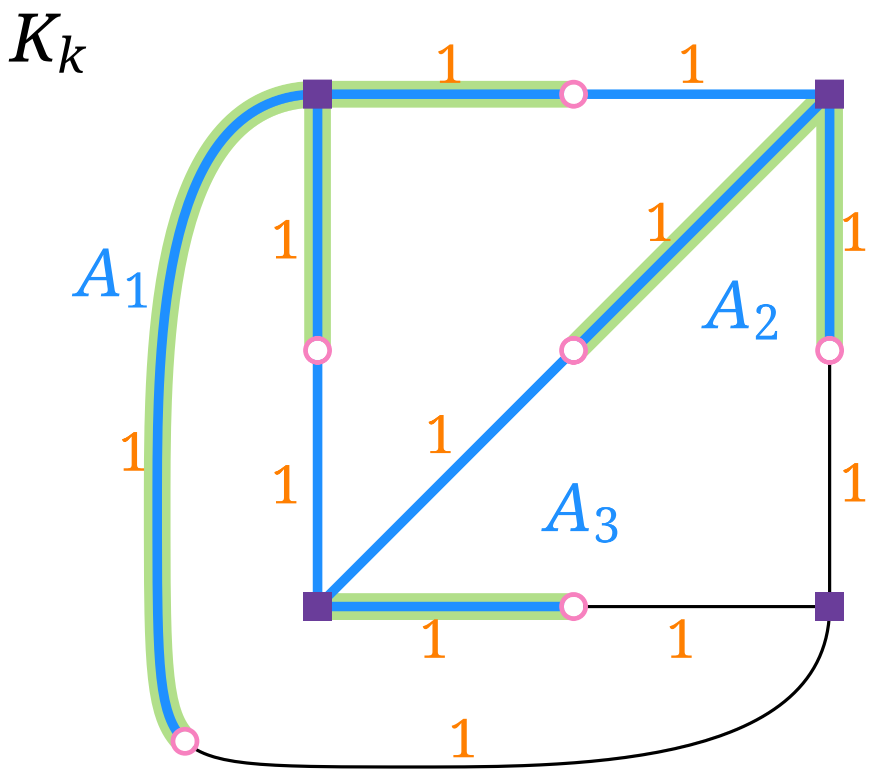
$$\mathrm{ALG} = (k-1)(k-1)$$

$$\mathrm{OPT} = \sum_{i=1}^{k-1} i = \frac{k\cdot(k-1)}{2}$$

$$\mathrm{ALG}/\mathrm{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$$

Can we do better?

Analysis Tight?



$\mathrm{ALG} = (k-1)(k-1)$

$\mathrm{OPT} = \sum_{i=1}^{k-1} i = \frac{k\cdot(k-1)}{2}$

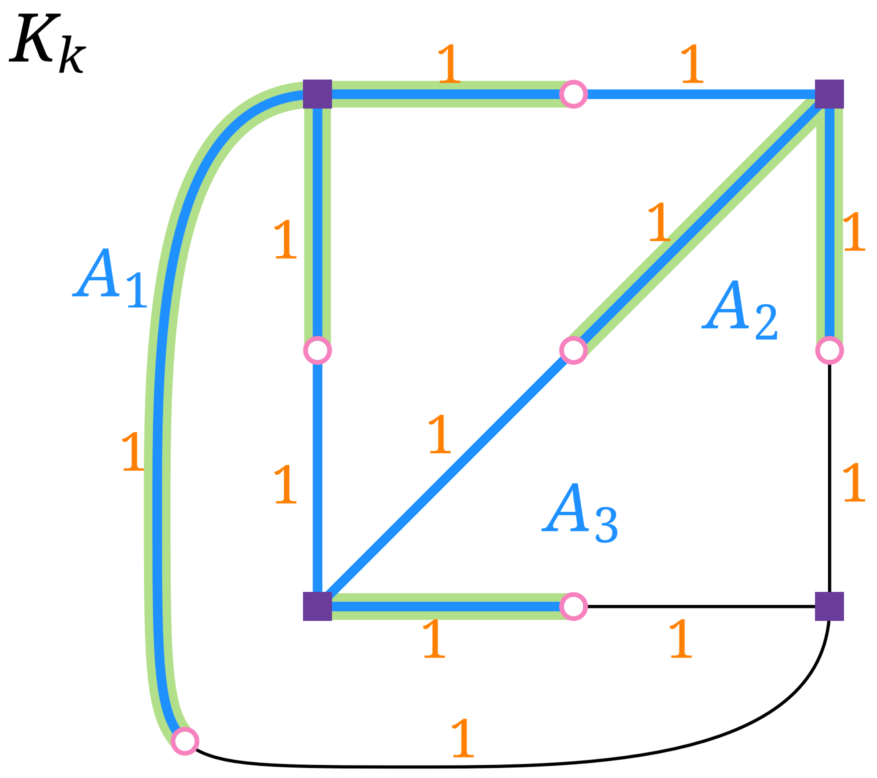
$\mathrm{ALG}/\mathrm{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$

Can we do better?

The best known approximation factor for MULTIWAYCUT is $1.2965 - \frac{1}{k}$.

[Sharma & Vondrák, STOC'14]

Analysis Tight?



$\text{ALG} = (k-1)(k-1)$

$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k\cdot(k-1)}{2}$

$\text{ALG}/\text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$

Can we do better?

The best known approximation factor for MULTIWAYCUT is $1.2965 - \frac{1}{k}$.

[Sharma & Vondrák, STOC'14]

MULTIWAYCUT cannot be approximated within factor $1.20016 - O(1/k)$ (unless P = NP).

[Bérczi, Chandrasekaran, Király & Madan, MP'18]

Summary

2-approximation for (Metric) Steiner Tree Problem and Metric TSP

ingredients: Eulerian tours, minimum spanning trees, triangle inequality (and Hamiltonian paths)

Summary

2-approximation for (Metric) Steiner Tree Problem and Metric TSP

ingredients: Eulerian tours, minimum spanning trees, triangle inequality (and Hamiltonian paths)

1.5-approximation for Metric TSP

additional ingredient: min-cost perfect matching (instead of Eulerian tour)

Summary

2-approximation for (Metric) Steiner Tree Problem and Metric TSP

ingredients: Eulerian tours, minimum spanning trees, triangle inequality (and Hamiltonian paths)

1.5-approximation for Metric TSP

additional ingredient: min-cost perfect matching (instead of Eulerian tour)

Approximation-Preserving Reduction

from Steiner tree problem to Metric Steiner Tree Problem

Summary

2-approximation for (Metric) Steiner Tree Problem and Metric TSP

ingredients: Eulerian tours, minimum spanning trees, triangle inequality (and Hamiltonian paths)

1.5-approximation for Metric TSP

additional ingredient: min-cost perfect matching (instead of Eulerian tour)

Approximation-Preserving Reduction

from Steiner tree problem to Metric Steiner Tree Problem

$(2 - 2/k)$-approximation algorithm for MultiwayCut

based on $k - 1$ isolating cuts realized by $s - t$-cuts