A single exponential FPT algorithm for the K_4-minor cover problem

Christophe Paul

CNRS - LIRMM, Montpellier France

Joint work with
EunJung Kim (CNRS - LIRMM)
Geevarghese Philip (IMSC, Chennai)

October 24, 2011

Parameterized K_4-Minor Cover

Given a graph $G = (V, E)$ and an integer k as parameter,

- is there a subset $S \subseteq V$ such that $G[V \setminus S]$ is K_4-minor free ?

Parameterized K_4-Minor Cover
(a.k.a. Parameterized Treewidth-two Vertex Deletion)

Given a graph $G = (V, E)$ and an integer k as parameter,

- is there a subset $S \subseteq V$ such that $G[V \setminus S]$ is K_4-minor free ?
- or is there a subset $S \subseteq V$ such that $tw(G[V \setminus S]) \leqslant 2$?

Parameterized K_4-Minor Cover
(a.k.a. Parameterized Treewidth-two Vertex Deletion)

Given a graph $G = (V, E)$ and an integer k as parameter,

- is there a subset $S \subseteq V$ such that $G[V \setminus S]$ is K_4-minor free ?
- or is there a subset $S \subseteq V$ such that $tw(G[V \setminus S]) \leqslant 2$?

Observations :

1. Vertex Cover $\equiv$ K_2-Minor Cover
   $\equiv$ Treewidth-zero Vertex Deletion

Parameterized K_4-Minor Cover
(a.k.a. Parameterized Treewidth-two Vertex Deletion)

Given a graph $G = (V, E)$ and an integer k as parameter,

- is there a subset $S \subseteq V$ such that $G[V \setminus S]$ is K_4-minor free ?
- or is there a subset $S \subseteq V$ such that $tw(G[V \setminus S]) \leqslant 2$?

Observations :

1. Vertex Cover $\equiv$ K_2-Minor Cover
   $\equiv$ Treewidth-zero Vertex Deletion

2. Feedback Vertex Set
   $\equiv$ K_3-Minor Cover
   $\equiv$ Treewidth-one Vertex Deletion

Parameterized K_4-Minor Cover
(a.k.a. Parameterized Treewidth-two Vertex Deletion)

Given a graph $G = (V, E)$ and an integer k as parameter,

- is there a subset $S \subseteq V$ such that $G[V \setminus S]$ is K_4-minor free ?
- or is there a subset $S \subseteq V$ such that $tw(G[V \setminus S]) \leqslant 2$?

Observations :

1. Vertex Cover $\equiv$ K_2-Minor Cover
   $\equiv$ Treewidth-zero Vertex Deletion

2. Feedback Vertex Set
   $\equiv$ K_3-Minor Cover
   $\equiv$ Treewidth-one Vertex Deletion

More generaly,

How fast can we solve Treewidth-t Vertex Deletion ?

Known results

1. Parameterized K_4-Minor Cover is FPT
   (by the Roberston and Seymour' graph minor theorem or by Courcelle's theorem)
2. Best algorithm runs is $2^{O(k \log k)}.n^{O(1)}$ [Fomin et al.'11]
3. No hope for a $2^{o(k)}.n^{O(1)}$ algorithm [???]

Known results

1. Parameterized K_4-Minor Cover is FPT
   (by the Roberston and Seymour' graph minor theorem or by Courcelle's theorem)
2. Best algorithm runs is $2^{O(k \log k)}.n^{O(1)}$ [Fomin et al.'11]
3. No hope for a $2^{o(k)}.n^{O(1)}$ algorithm [???]

Our result

There exists an algorithm that solves the Parameterized K_4-Minor Cover problem in time $2^{O(k)}.n^{O(1)}$.

Known results

1. Parameterized K_4-Minor Cover is FPT
   (by the Roberston and Seymour' graph minor theorem or by Courcelle's theorem)
2. Best algorithm runs is $2^{O(k \log k)}.n^{O(1)}$ [Fomin et al.'11]
3. No hope for a $2^{o(k)}.n^{O(1)}$ algorithm [???]

Our result

There exists an algorithm that solves the Parameterized K_4-Minor Cover problem in time $2^{O(k)}.n^{O(1)}$.

Ingredients:

- iterative compression,
- protrusions, reduction and branching rules,
- Vertex Cover in circle graphs

Iterative compression for Feedback Vertex Set as a warm-up

Disjoint-K_4-Minor Cover

- Reduction Rules
- Protrusion Rule
- Branching Rules

Algorithm for the Disjoint-K_4-Minor Cover

Disjoint-Feedback Vertex Set (Disjoint-FVS)

- Given a graph $G = (V, E)$ a feedback vertex set S of size k
- Compute (if it exists) a feedback vertex set S' such that $S \cap S' = \emptyset$ and $|S'| < k$

Disjoint-Feedback Vertex Set (Disjoint-FVS)

- Given a graph $G = (V, E)$ a feedback vertex set S of size k
- Compute (if it exists) a feedback vertex set S' such that $S \cap S' = \emptyset$ and $|S'| < k$

Lemma

A single exponential FPT algorithm for Disjoint-FVS
$\Rightarrow$ a single exponential FPT algorithm for FVS.

Disjoint-Feedback Vertex Set (Disjoint-FVS)

- Given a graph $G = (V, E)$ a feedback vertex set S of size k
- Compute (if it exists) a feedback vertex set S' such that $S \cap S' = \emptyset$ and $|S'| < k$

Lemma

A single exponential FPT algorithm for Disjoint-FVS
$\Rightarrow$ a single exponential FPT algorithm for FVS.

We use

- branching and reduction rules
- a measure function to analyse the time complexity

Skip example

Disjoint-Feedback Vertex Set (Disjoint-FVS)

- Given a graph $G = (V, E)$ a feedback vertex set S of size k
- Compute (if it exists) a feedback vertex set S' such that $S \cap S' = \emptyset$ and $|S'| < k$

Lemma

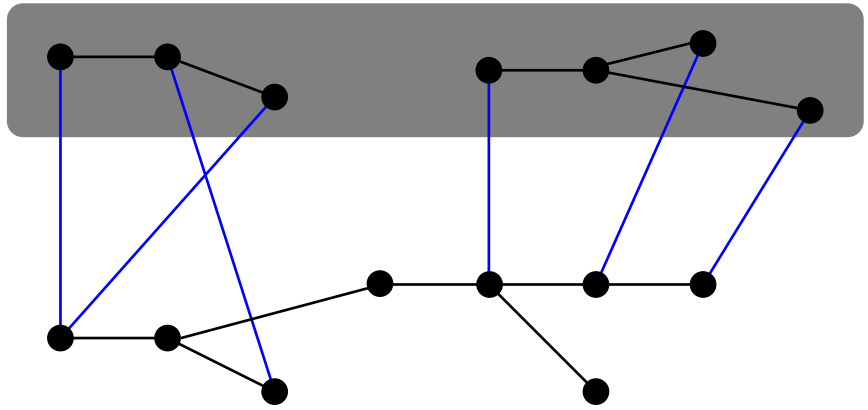
A single exponential FPT algorithm for Disjoint-FVS
$\Rightarrow$ a single exponential FPT algorithm for FVS.

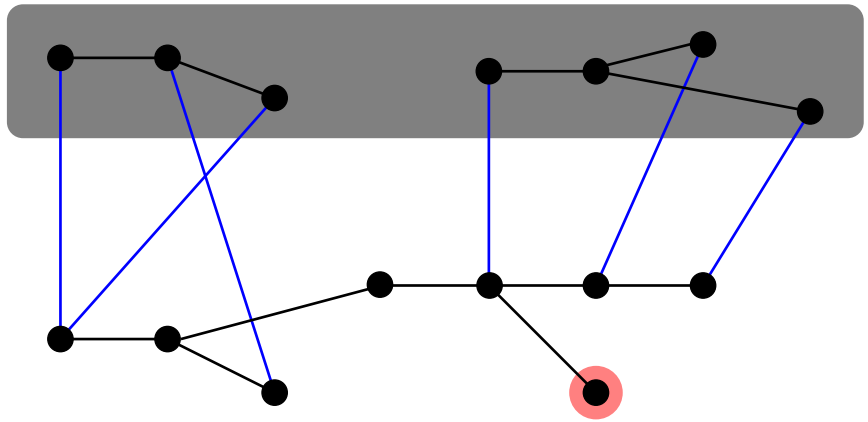
We use

- branching and reduction rules
- a measure function to analyse the time complexity

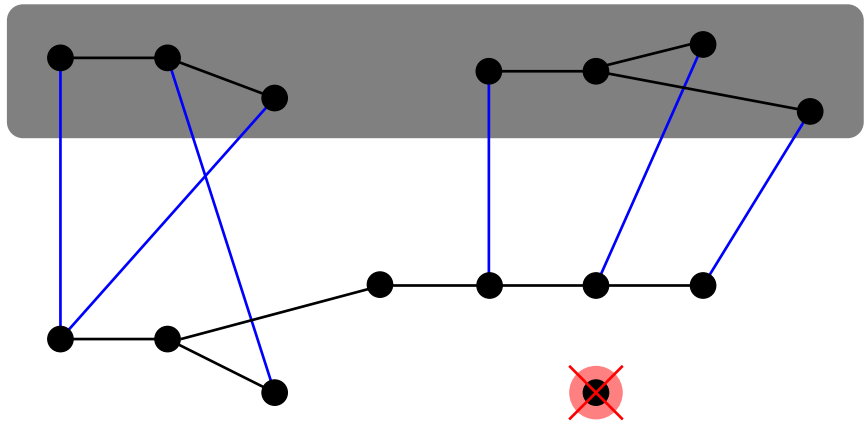
$$\boxed{\mu = k + \#cc(G[S])}$$

Skip example

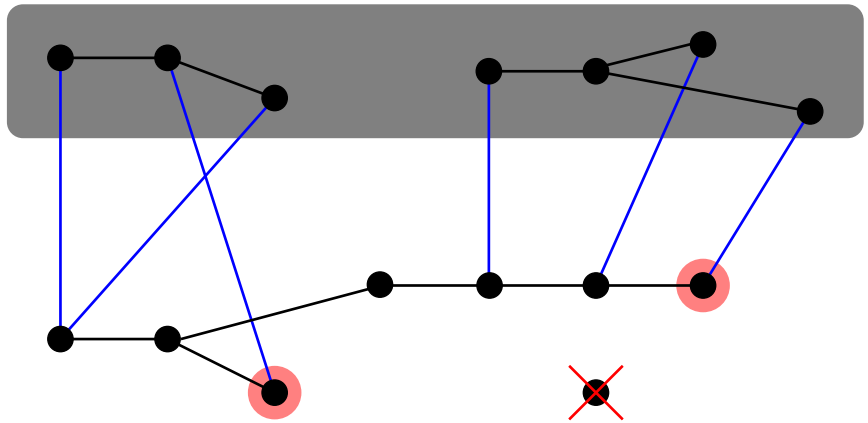




Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

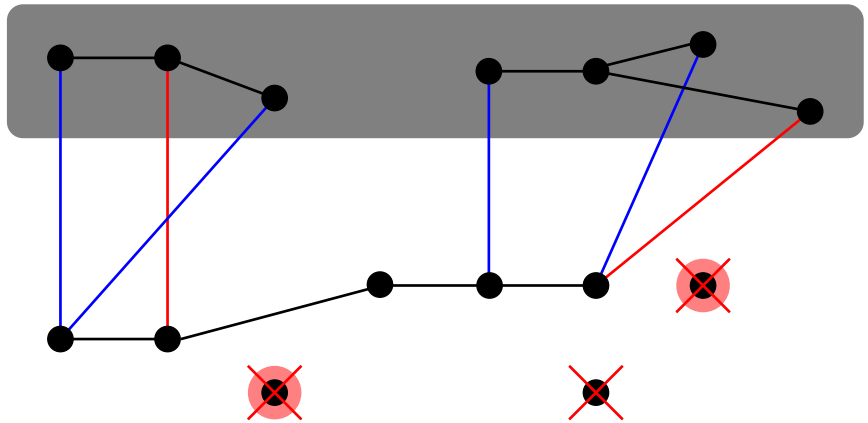


Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$



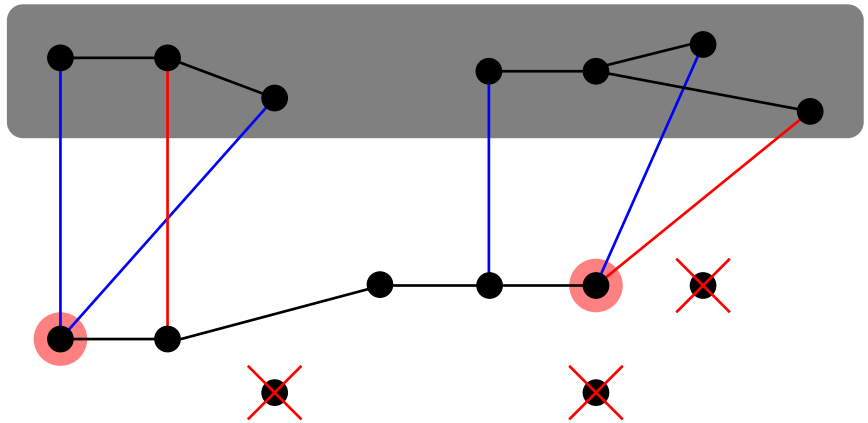
Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

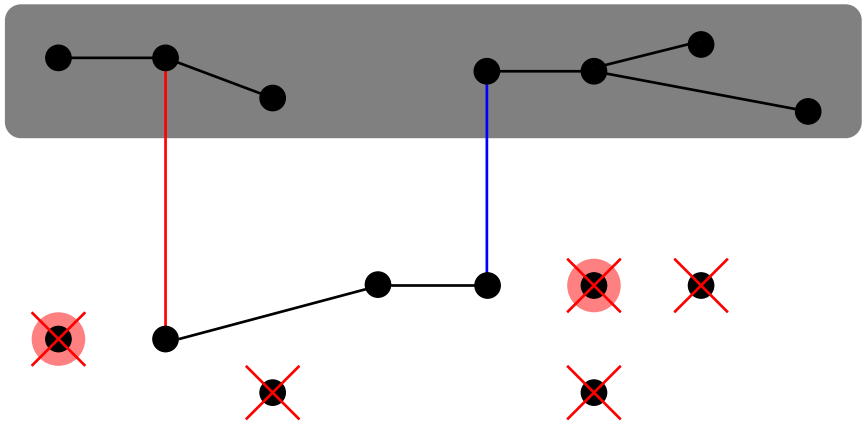
Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

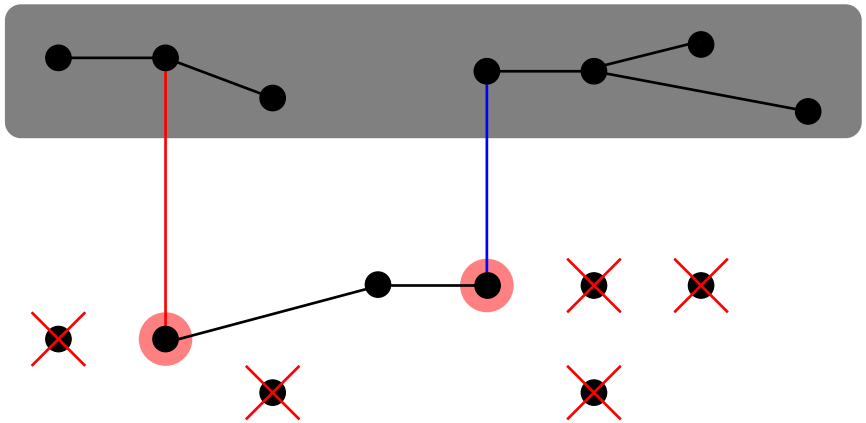
Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

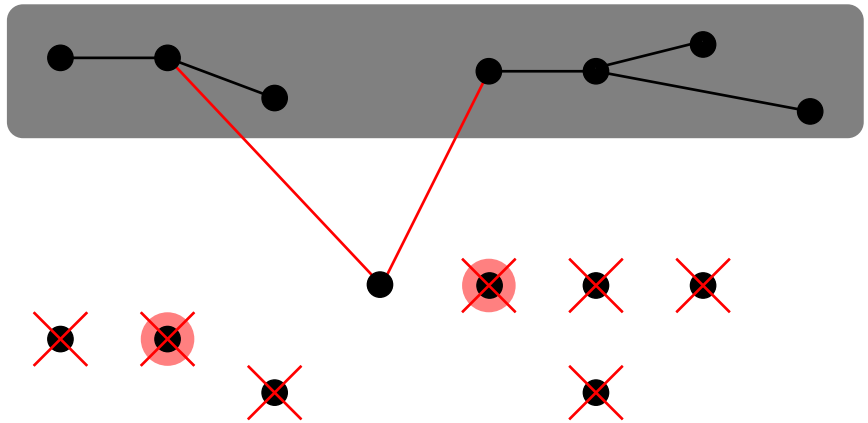
Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

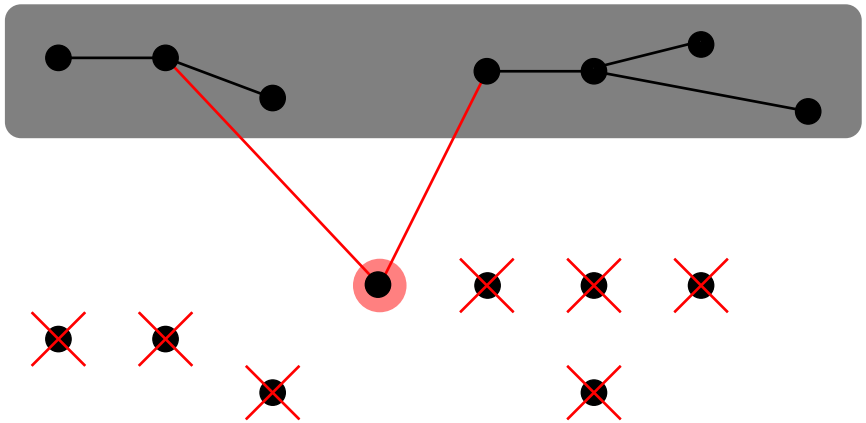
Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1



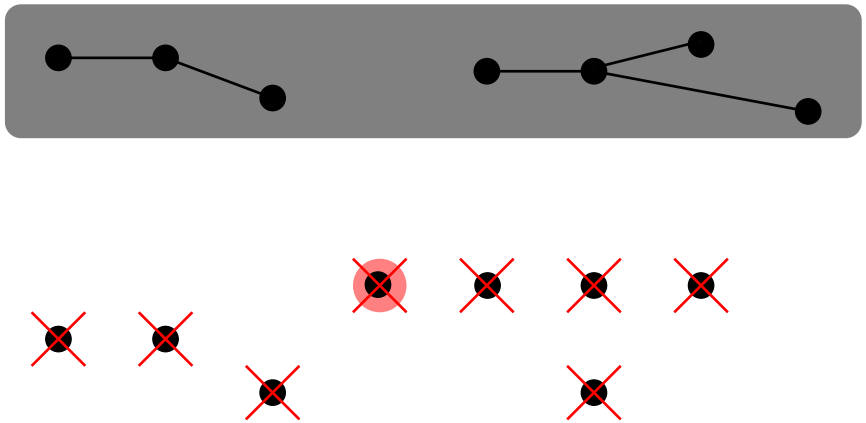
Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1

Branching Rule: If $x \in V \setminus S$ has two neighbours in two different connected components of $G[S]$, then branch on

- $(G - \{x\}, S, k - 1)$ $\Rightarrow \mu$ decreases



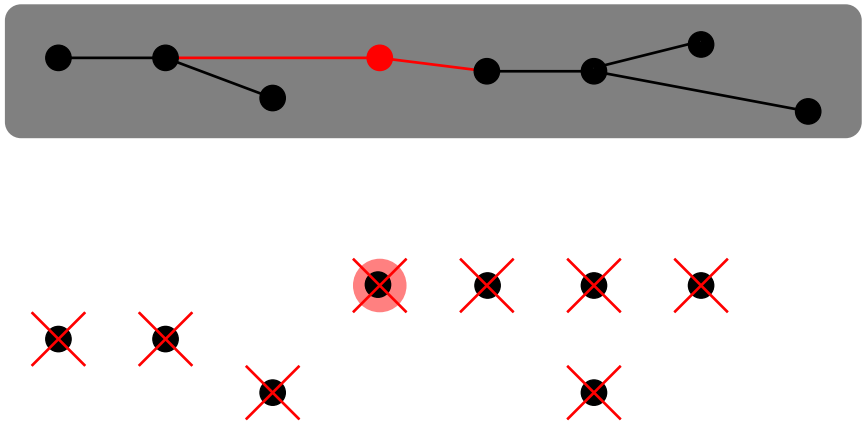
Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1

Branching Rule: If $x \in V \setminus S$ has two neighbours in two different connected components of $G[S]$, then branch on

- $(G - \{x\}, S, k - 1)$ $\Rightarrow \mu$ decreases



Red. Rule 1: Remove leaf $x \in V \setminus S$ if $N(x) \cap S = \emptyset$

Red. Rule 2: Bypass leaf $x \in V \setminus S$ if $d(x) = 2$, $|N(x) \cap S| = 1$

Red. Rule 3: Remove every vertex $x \in V \setminus S$ with at least 2 neighbours in some connect. comp. C of $G[S]$ and decrease k by 1

Branching Rule: If $x \in V \setminus S$ has two neighbours in two different connected components of $G[S]$, then branch on

- $(G - \{x\}, S, k - 1)$ $\Rightarrow$ μ decreases
- $(G, S \cup \{x\}, k)$ $\Rightarrow$ μ decreases

Lemma

Disjoint-FVS can be solved in single exponential time: $4^k.n^{O(1)}$

Lemma

DISJOINT-FVS can be solved in single exponential time: $4^k.n^{O(1)}$

- measure function μ is bounded by $2k$
- branching degree is 2

Lemma

Disjoint-FVS can be solved in single exponential time: $4^k . n^{O(1)}$

- measure function μ is bounded by $2k$
- branching degree is 2

For the Disjoint-K_4-Minor Cover we need to

- adapt the measure function μ
- adapt the reduction and branching rules

Lemma

DISJOINT-FVS can be solved in single exponential time: $4^k.n^{O(1)}$

- measure function μ is bounded by $2k$
- branching degree is 2

For the DISJOINT-K_4-MINOR COVER we need to

- adapt the measure function μ
- adapt the reduction and branching rules

Our algorithm deeply relies on:

Lemma: A graph is K_4-minor free iff
its biconnected components are series-parallel graphs

Lemma

DISJOINT-FVS can be solved in single exponential time: $4^k . n^{O(1)}$

- measure function μ is bounded by $2k$
- branching degree is 2

For the DISJOINT-K_4-MINOR COVER we need to

- adapt the measure function μ
- adapt the reduction and branching rules

Our algorithm deeply relies on:

Lemma: A graph is K_4-minor free iff
its biconnected components are series-parallel graphs

$\Rightarrow$ Obs.: biconnected comp. of $G[S]$ and $G[V \setminus S]$ are SP-graphs

Iterative compression for Feedback Vertex Set as a warm-up

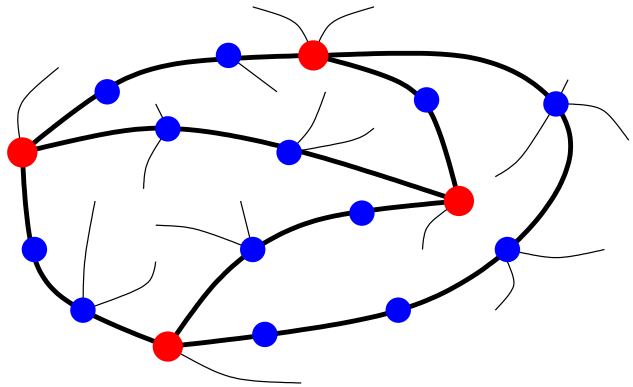
Disjoint-K_4-Minor Cover

- Reduction Rules
- Protrusion Rule
- Branching Rules

Algorithm for the Disjoint-K_4-Minor Cover

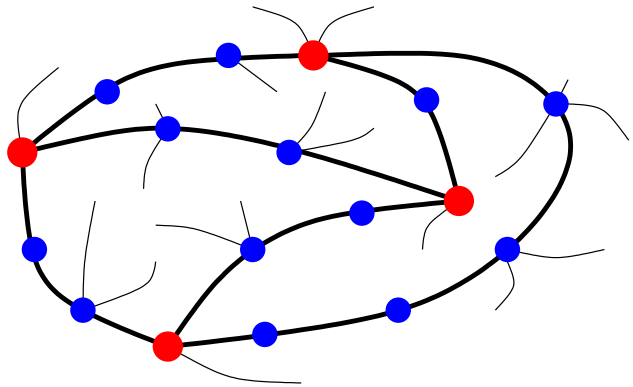
Some terminology

Observation: A graph contains a K_4-minor iff it contains a K_4-subdivision.



Some terminology

Observation: A graph contains a K_4-minor iff it contains a K_4-subdivision.

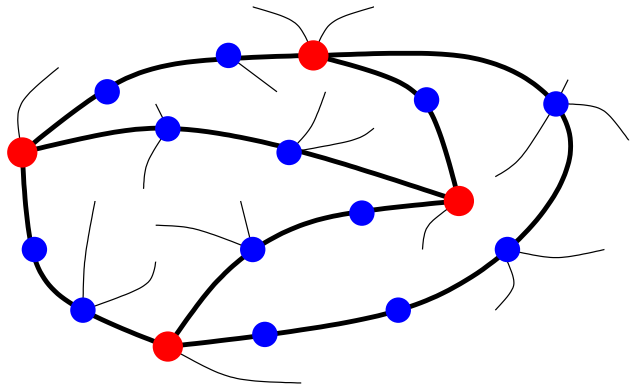


In a K_4-subdivision

- Degree 3 vertices are called branching nodes
- Degree 2 vertices are called subdivision nodes

Some terminology

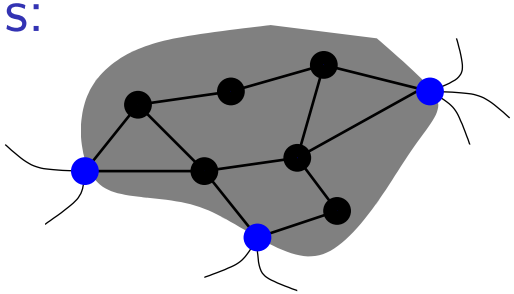
Observation: A graph contains a K_4-minor iff it contains a K_4-subdivision.



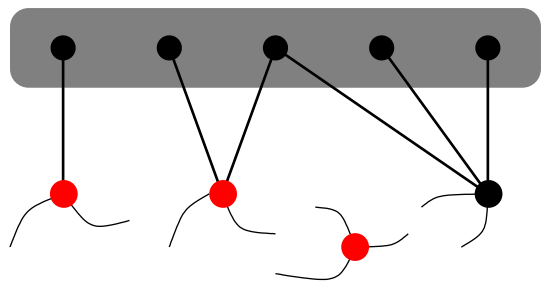
In a K_4-subdivision

- Degree 3 vertices are called branching nodes
- Degree 2 vertices are called subdivision nodes

Notations:

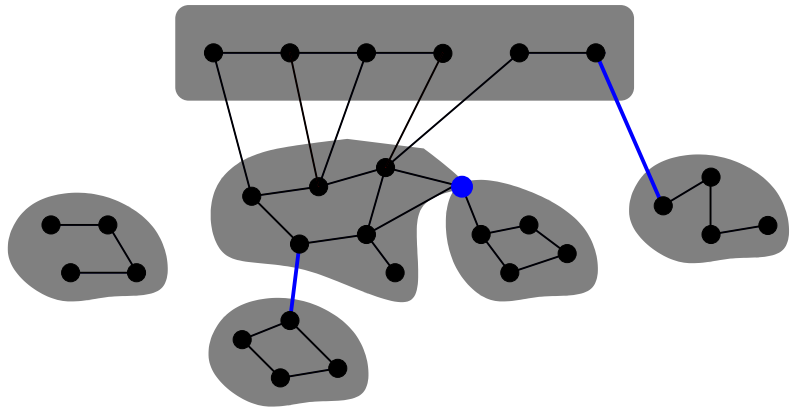


$\partial_G(X)$



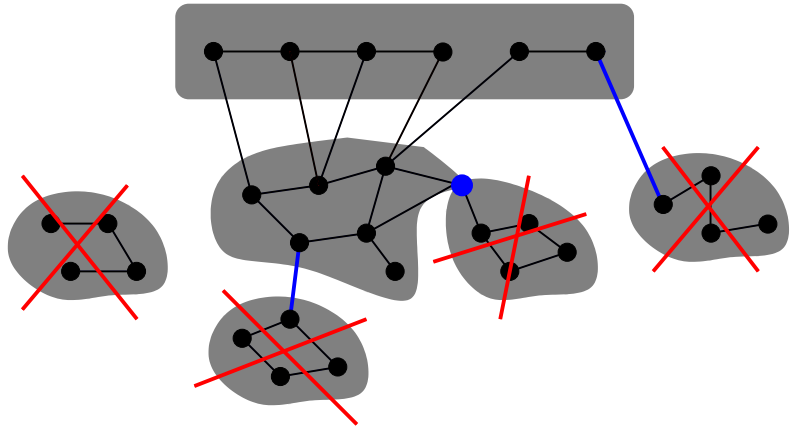
$N_0,\ N_1,\ N_2$

Reduction Rules (1)



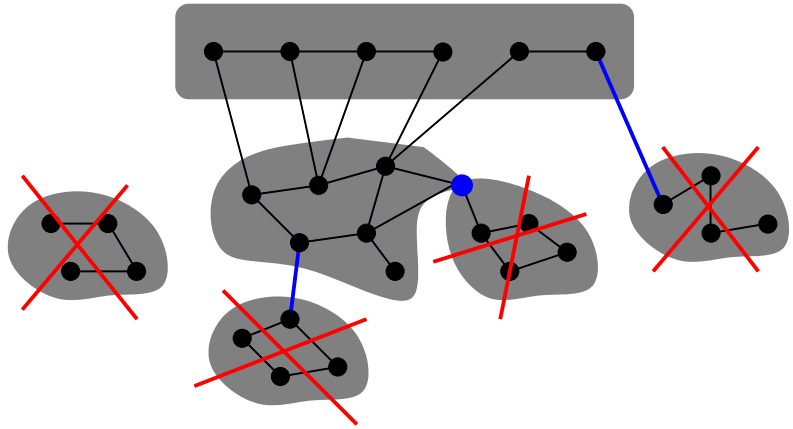
Reduction rule 1: Let $X \subseteq V \setminus S$.
If $G[X]$ is a connected component of G or if G contains a cut edge disconnecting X, then
If G has a cut vertex x disconnecting X, then

Reduction Rules (1)



Reduction rule 1: Let $X \subseteq V \setminus S$.
If $G[X]$ is a connected component of G or if G contains a cut edge disconnecting X, then remove X.
If G has a cut vertex x disconnecting X, then remove $X \setminus \{x\}$.

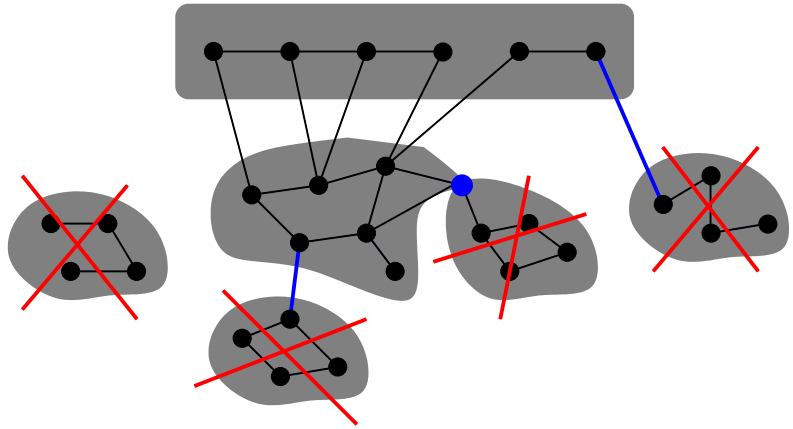
Reduction Rules (1)



Reduction rule 1: Let $X \subseteq V \setminus S$.
If $G[X]$ is a connected component of G or if G contains a cut edge disconnecting X, then remove X.
If G has a cut vertex x disconnecting X, then remove $X \setminus \{x\}$.

- "No" K_4-subdivision involves vertices of X

Reduction Rules (1)



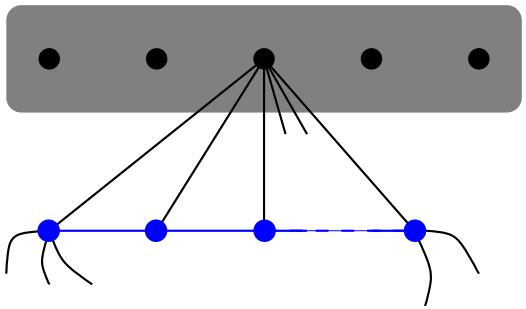
Reduction rule 1: Let $X \subseteq V \setminus S$.
If $G[X]$ is a connected component of G or if G contains a cut edge disconnecting X, then remove X.
If G has a cut vertex x disconnecting X, then remove $X \setminus \{x\}$.

- "No" K_4-subdivision involves vertices of X

Reduction Rule 2: Bypass every vertex of degree 2 in G with one neighbour in S and the other in $V \setminus S$.

Reduction Rule 3: Transform multiple edges into single edge.

Reduction Rules (2)

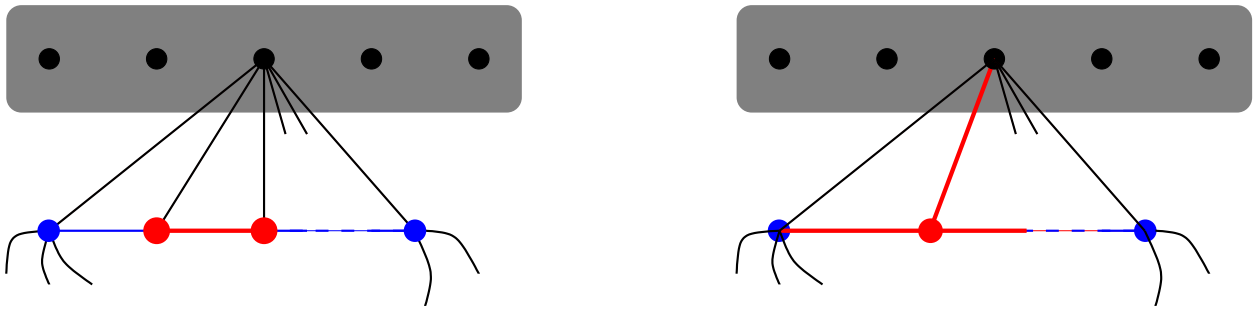


Reduction Rule 4: Let $X = \{x_1, x_2, x_3 \dots x_l\} \subseteq V \setminus S$ be a set of $l \geqslant 4$ vertices inducing a path in $G[V \setminus S]$ such that

- $\forall i \in [l]$, $N(x_i) \cap S = \{v\}$
- $\forall i \in [2, l-1]$, x_i has degree 3

Then

Reduction Rules (2)

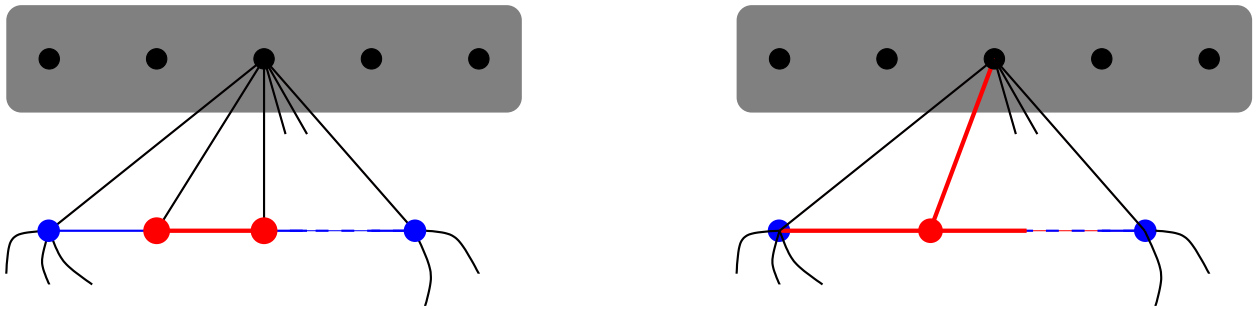


Reduction Rule 4: Let $X = \{x_1, x_2, x_3 \dots x_l\} \subseteq V \setminus S$ be a set of $l \geqslant 4$ vertices inducing a path in $G[V \setminus S]$ such that

- $\forall i \in [l]$, $N(x_i) \cap S = \{v\}$
- $\forall i \in [2, l-1]$, x_i has degree 3

Then contract the edge (x_2, x_3) and remove the new multiple edge.

Reduction Rules (2)



Reduction Rule 4: Let $X = \{x_1, x_2, x_3 \dots x_l\} \subseteq V \setminus S$ be a set of $l \geqslant 4$ vertices inducing a path in $G[V \setminus S]$ such that

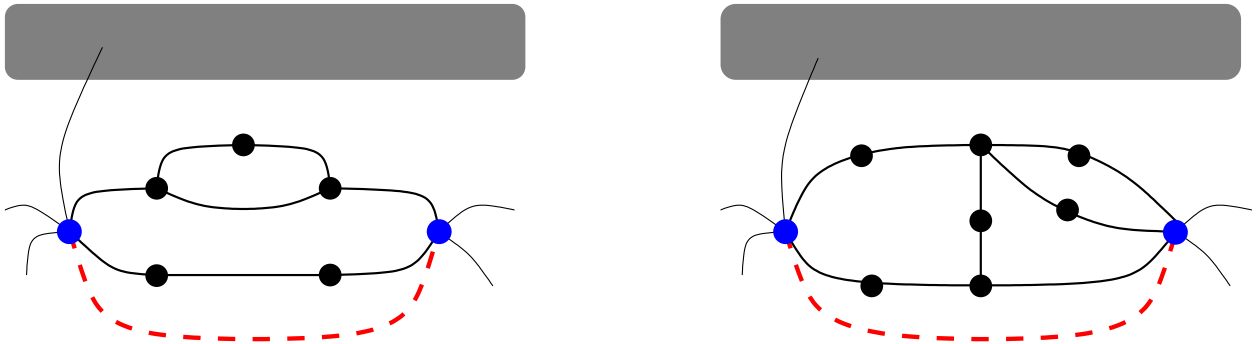
- $\forall i \in [l]$, $N(x_i) \cap S = \{v\}$
- $\forall i \in [2, l-1]$, x_i has degree 3

Then contract the edge (x_2, x_3) and remove the new multiple edge.

- X contains at most 3 branching nodes of a K_4-subdivision.

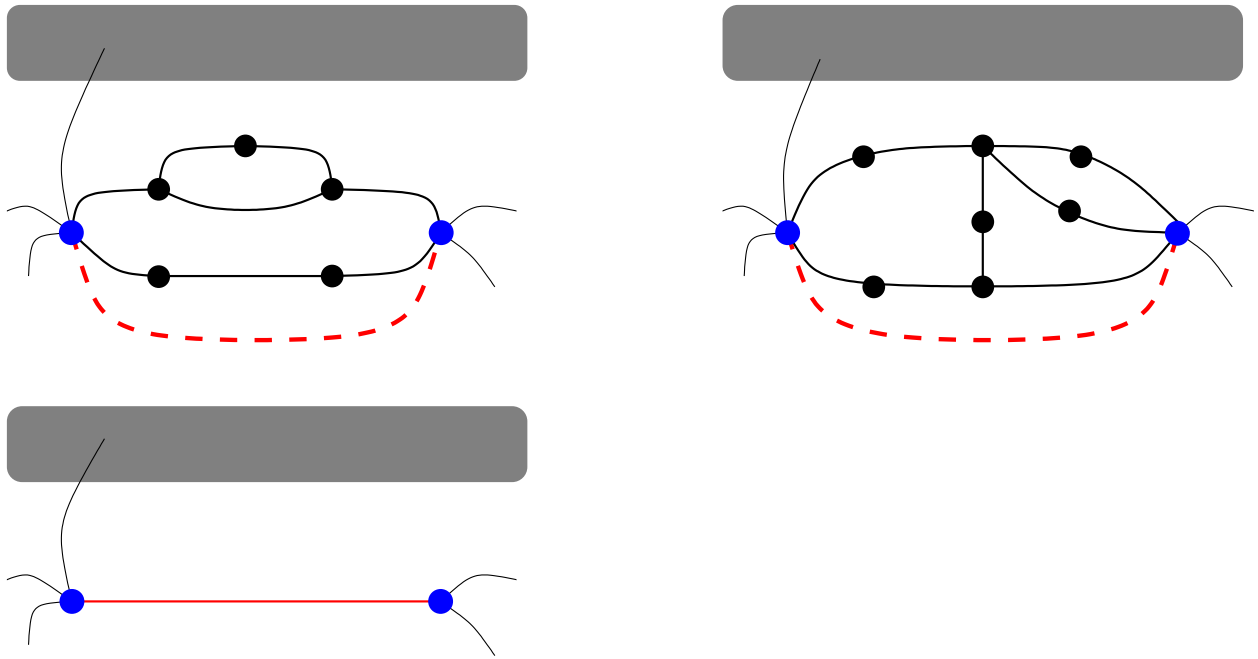
  $\Rightarrow$ So one of its vertices is used as a subdivision node.

Reduction Rules (3)



Reduction Rule 5: Let $X \subseteq V \setminus S$ be such that $\partial_G(X) = \{s, t\}$ and $X \setminus \{x, y\} \subseteq N_0$. Then

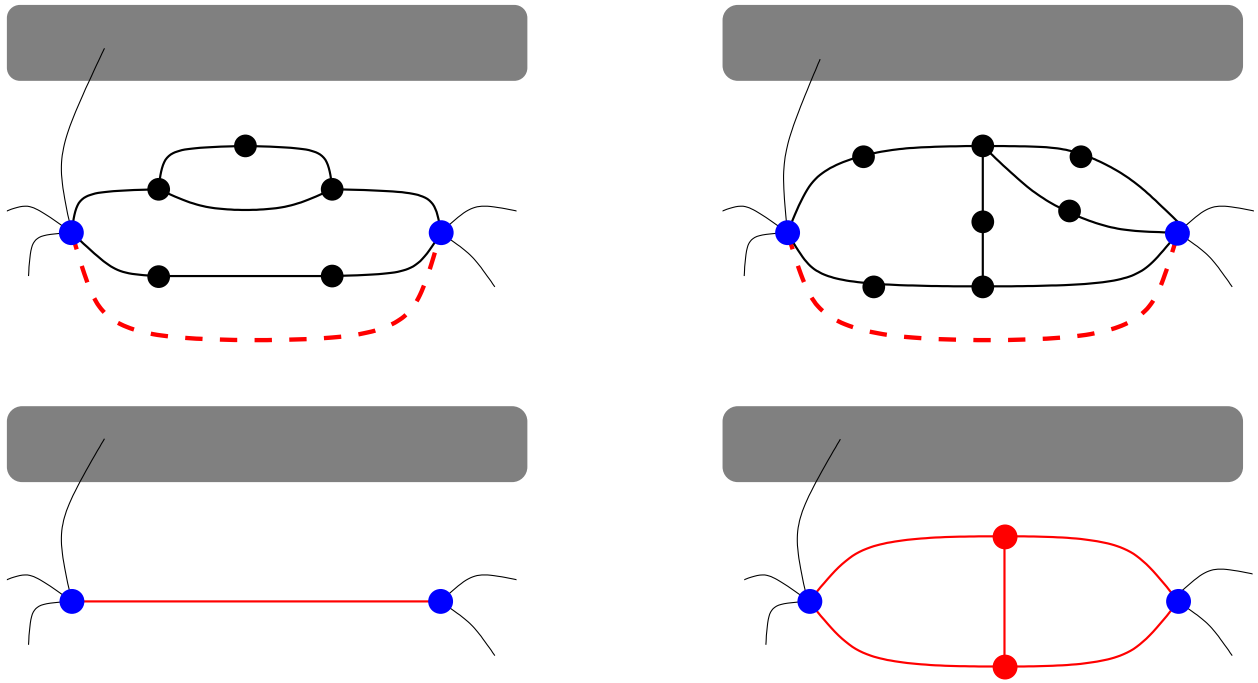
Reduction Rules (3)



Reduction Rule 5: Let $X \subseteq V \setminus S$ be such that $\partial_G(X) = \{s, t\}$ and $X \setminus \{x, y\} \subseteq N_0$. Then

- if $G[X] + (s, t)$ is K_4-minor free, replace $G[X]$ by the edge (s, t)

Reduction Rules (3)



Reduction Rule 5: Let $X \subseteq V \setminus S$ be such that $\partial_G(X) = \{s, t\}$ and $X \setminus \{x, y\} \subseteq N_0$. Then

- if $G[X] + (s, t)$ is K_4-minor free, replace $G[X]$ by the edge (s, t)
- otherwise, replace $G[X]$ by a 4-cycle containing s and t with a chord not incident to s nor t.

Iterative compression for Feedback Vertex Set as a warm-up

Disjoint-K_4-Minor Cover

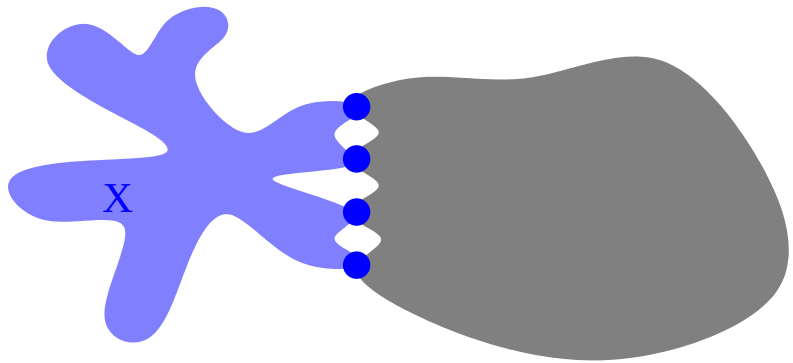
- Reduction Rules
- Protrusion Rule
- Branching Rules

Algorithm for the Disjoint-K_4-Minor Cover

Protrusion Rule (1)

A t-protrusion in a graph $G = (V, E)$ is a subset $X \subseteq V$ such that

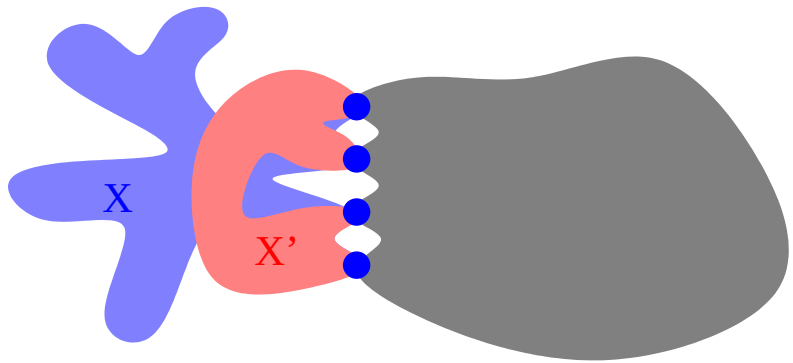
$$tw(G[X]) \leqslant t \text{ and } |\partial_G(X)| \leqslant t$$



Protrusion Rule (1)

A t-protrusion in a graph $G = (V, E)$ is a subset $X \subseteq V$ such that

$$tw(G[X]) \leqslant t \text{ and } |\partial_G(X)| \leqslant t$$

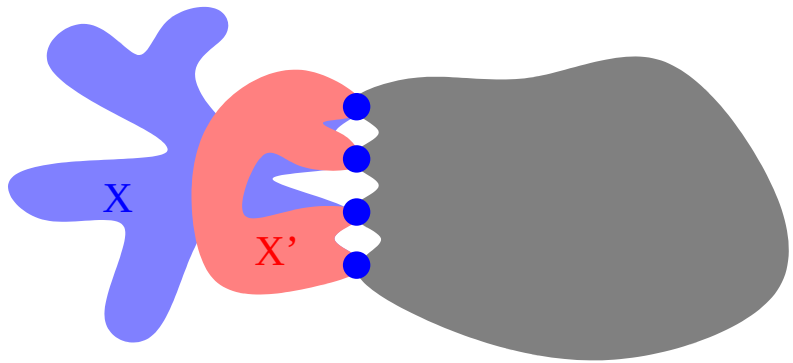


Idea of a protrusion rule: replace a t-protrusion X with a smaller t-protrusion X' while preserving the instance equivalence

Protrusion Rule (1)

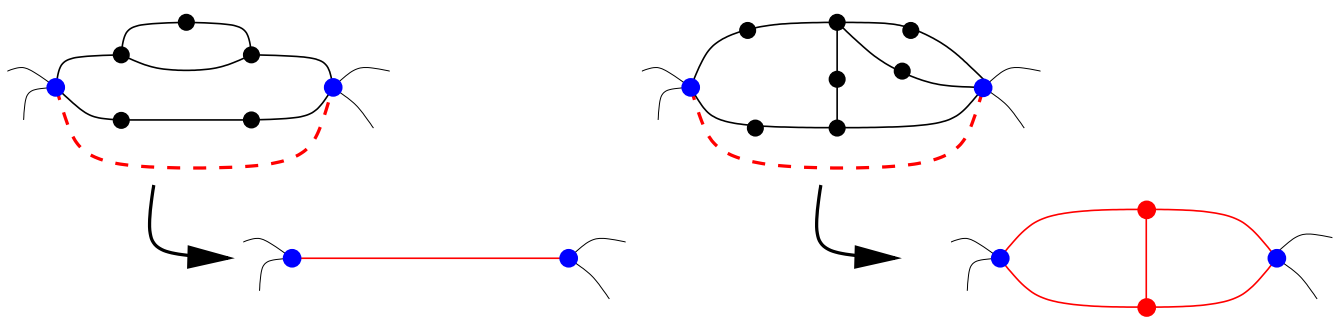
A t-protrusion in a graph $G = (V, E)$ is a subset $X \subseteq V$ such that

$$tw(G[X]) \leqslant t \text{ and } |\partial_G(X)| \leqslant t$$



Idea of a protrusion rule: replace a t-protrusion X with a smaller t-protrusion X' while preserving the instance equivalence

Observation: Reduction rule 5 is a example of protrusion rule

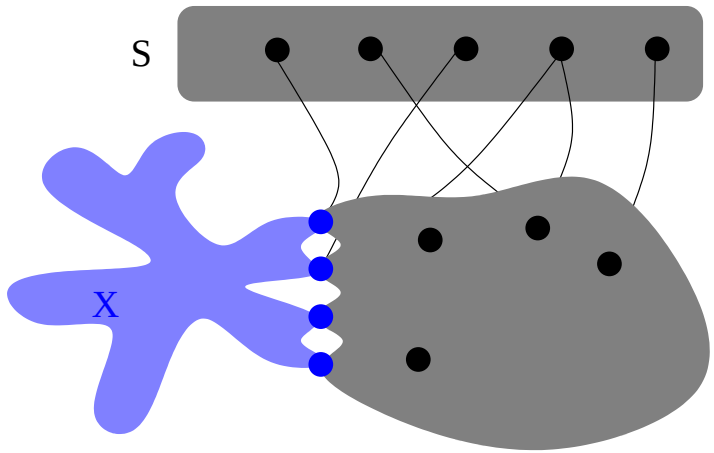


Protrusion Rule (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover

Protrusion Rule: Let X be a t-protrusion of G such that

$$X \cap S = \emptyset \text{ and } |X| > \gamma(t)$$



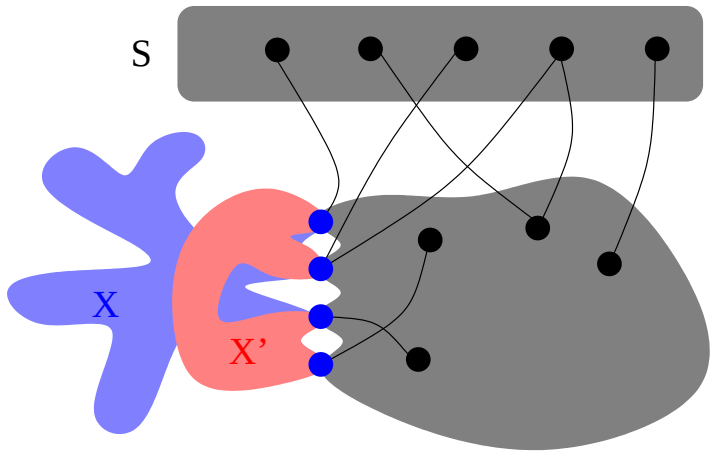
Then,

Protrusion Rule (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover

Protrusion Rule: Let X be a t-protrusion of G such that

$$X \cap S = \emptyset \text{ and } |X| > \gamma(t)$$



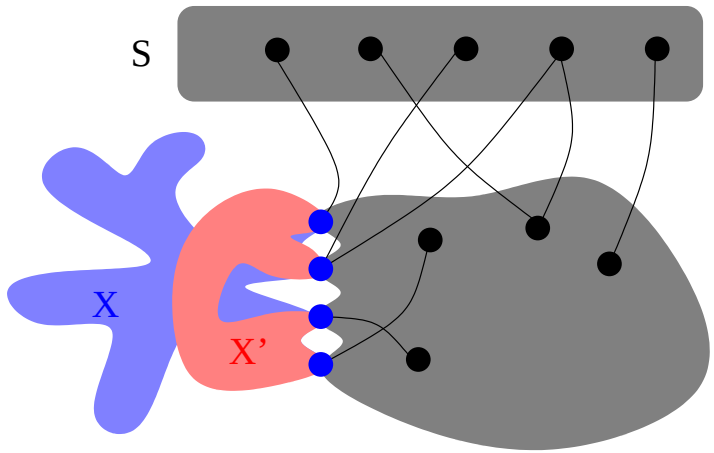
Then, replaces X with a t-protrusion X' to get (G', S, k') st.

Protrusion Rule (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover

Protrusion Rule: Let X be a t-protrusion of G such that

$$X \cap S = \emptyset \text{ and } |X| > \gamma(t)$$



Then, replaces X with a t-protrusion X' to get (G', S, k') st.

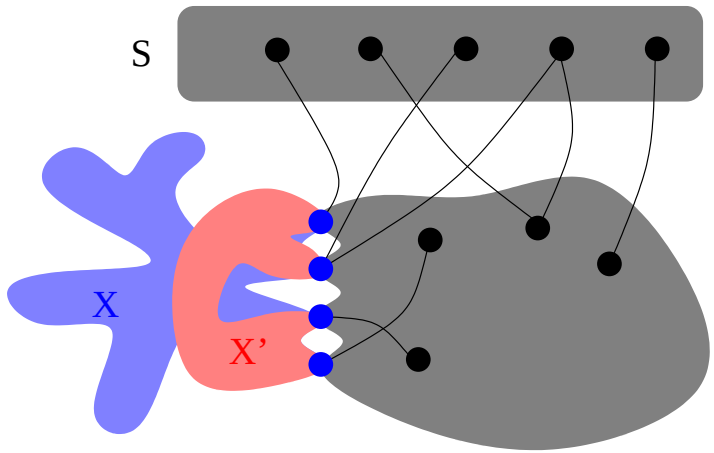
- $|X'| < |X|$, $k' \leqslant k$,

Protrusion Rule (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover

Protrusion Rule: Let X be a t-protrusion of G such that

$$X \cap S = \emptyset \text{ and } |X| > \gamma(t)$$



Then, replaces X with a t-protrusion X' to get (G', S, k') st.

- $|X'| < |X|$, $k' \leqslant k$,
- $G'[S]$ and $G[S]$ are isomorphic and $G' - S'$ is K_4-minor free

Iterative compression for Feedback Vertex Set as a warm-up

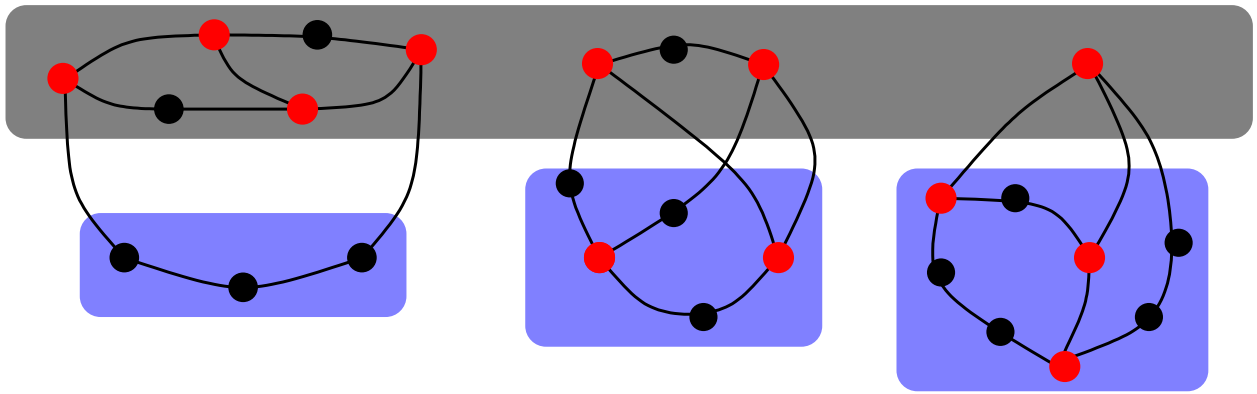
Disjoint-K_4-Minor Cover

- Reduction Rules
- Protrusion Rule
- Branching Rules

Algorithm for the Disjoint-K_4-Minor Cover

Branching Rules (1)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.

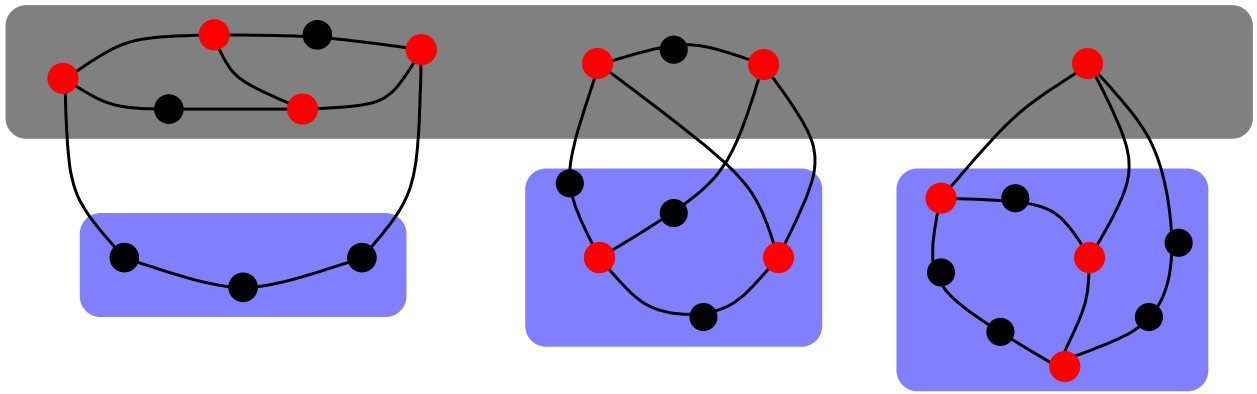


Branching Rule 1: If $X \subseteq V \setminus S$ is a set such that $G[S \cup X]$ contains a K_4-subdivision, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$.

Branching Rules (1)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.



Branching Rule 1: If $X \subseteq V \setminus S$ is a set such that $G[S \cup X]$ contains a K_4-subdivision, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$.

Observation: To bound the branching degree, we have to guarantee that the above rule always applies on a set X such that $|X| \leqslant c$ for some constant c.

Branching Rules (2)

We introduce a new measure function:

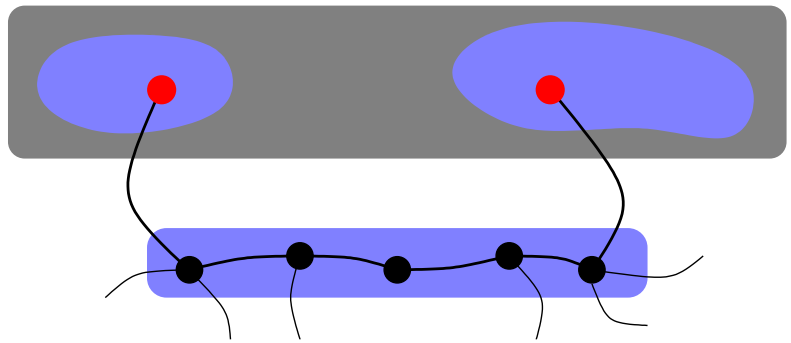
$$\mu = c_1 \times k + c_1 \times \#cc(G[S]) + \#bc(G[S])$$

Branching Rules (2)

We introduce a new measure function:

$$\boxed{\mu = c_1 \times k + c_1 \times \#cc(G[S]) + \#bc(G[S])}$$

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.



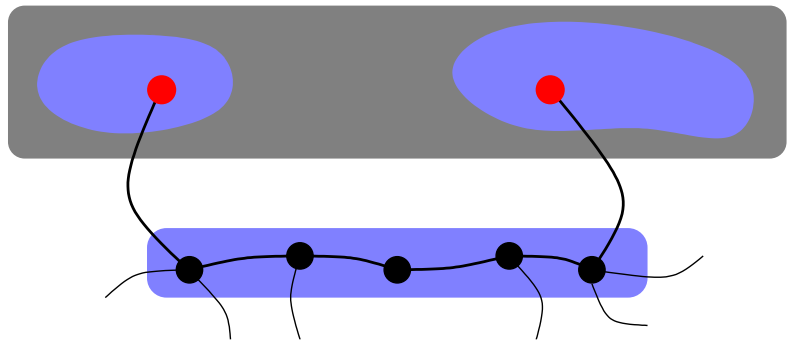
Branching Rule 2: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) \neq cc_S(s_2)$, then branch on

Branching Rules (2)

We introduce a new measure function:

$$\boxed{\mu = c_1 \times k + c_1 \times \#cc(G[S]) + \#bc(G[S])}$$

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.

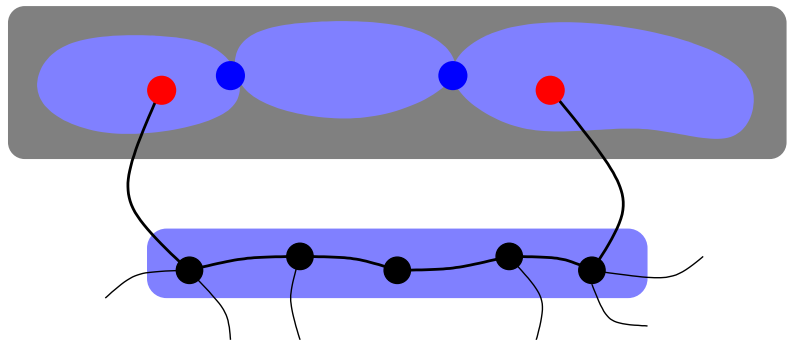


Branching Rule 2: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) \neq cc_S(s_2)$, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$
- $(G, S \cup X, k)$.

Branching Rules (2)

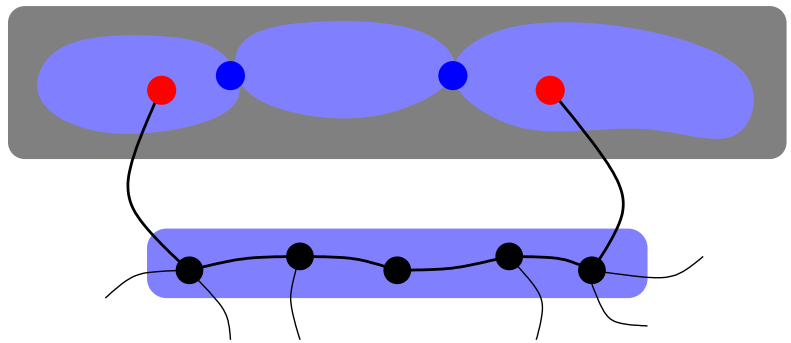
Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.



Branching Rule 3: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) = cc_S(s_2)$ and $bc_S(s_1) \neq bc_S(s_2)$, then branch on

Branching Rules (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.

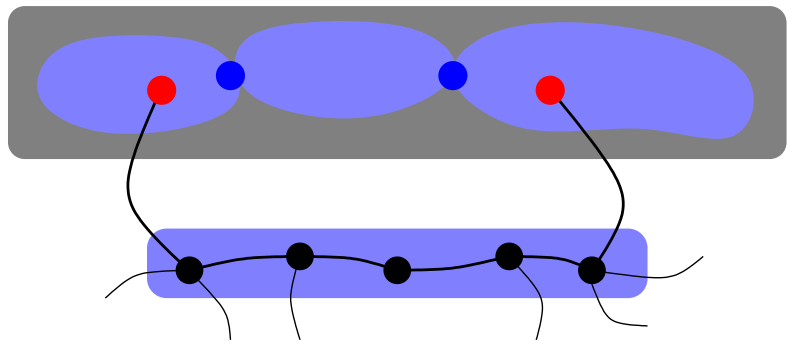


Branching Rule 3: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) = cc_S(s_2)$ and $bc_S(s_1) \neq bc_S(s_2)$, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$
- $(G, S \cup X, k)$.

Branching Rules (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.



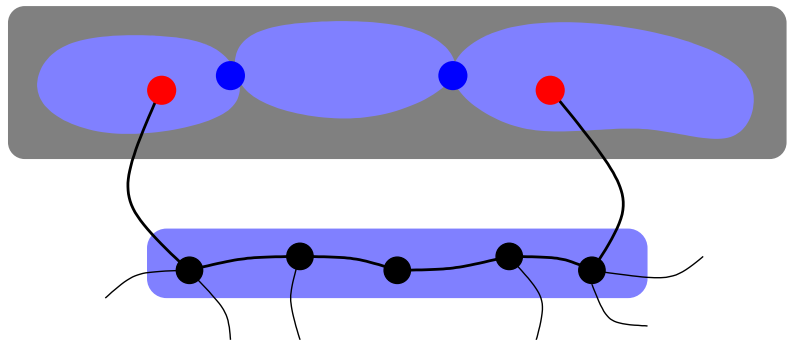
Branching Rule 3: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) = cc_S(s_2)$ and $bc_S(s_1) \neq bc_S(s_2)$, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$
- $(G, S \cup X, k)$.

Claim: $\mu = c_1 \times k + c_1 \times \#cc(G[S]) + \#bc(G[S])$ is decreasing

Branching Rules (2)

Let (G, S, k) be an instance of Disjoint-K_4-Minor Cover.



Branching Rule 3: If $X \subseteq V \setminus S$ is connected and $\{s_1, s_2\} \subseteq N_S(X)$ with $cc_S(s_1) = cc_S(s_2)$ and $bc_S(s_1) \neq bc_S(s_2)$, then branch on

- $(G - \{x\}, S, k - 1)$ for every $x \in X$
- $(G, S \cup X, k)$.

Claim: $\mu = c_1 \times k + c_1 \times \#cc(G[S]) + \#bc(G[S])$ is decreasing

- c_1 depends on the maximum size of the sets X on which the branching rules is applies

Iterative compression for Feedback Vertex Set as a warm-up

Disjoint-K_4-Minor Cover

- Reduction Rules
- Protrusion Rule
- Branching Rules

Algorithm for the Disjoint-K_4-Minor Cover

Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

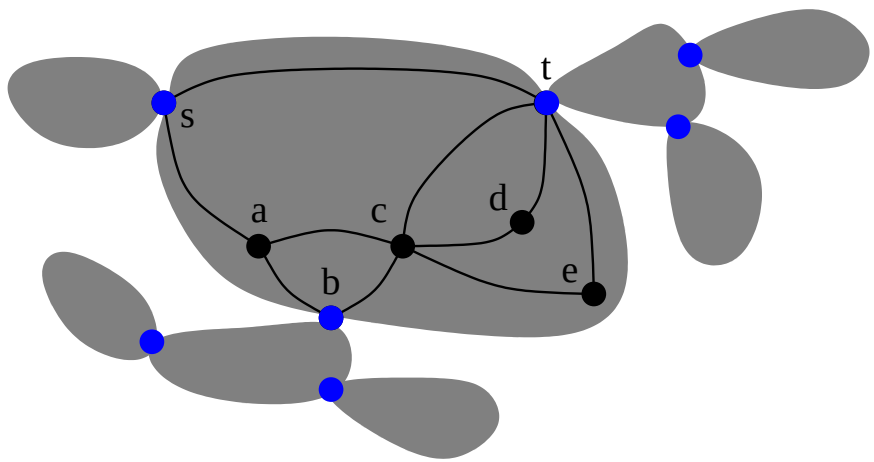
   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition

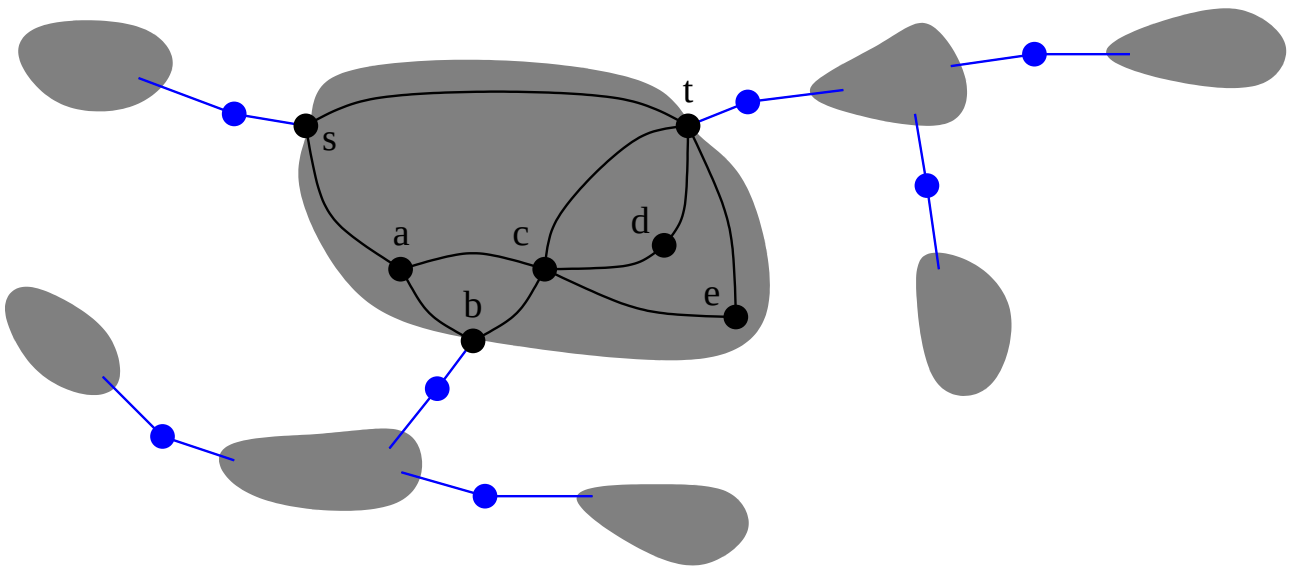


Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition

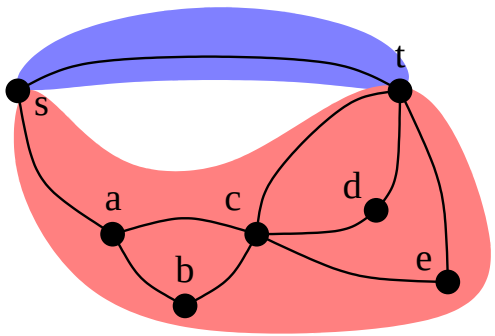


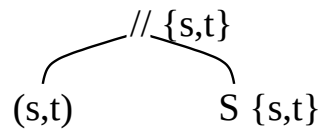
Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition



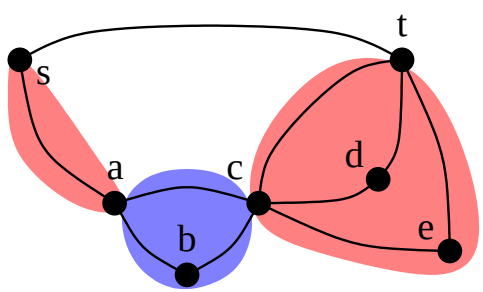


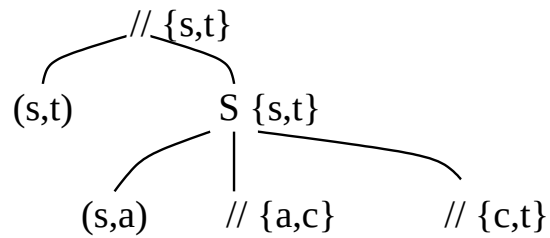
Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition



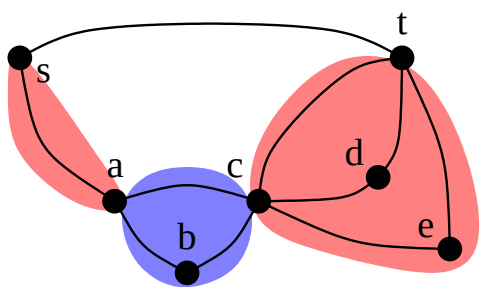


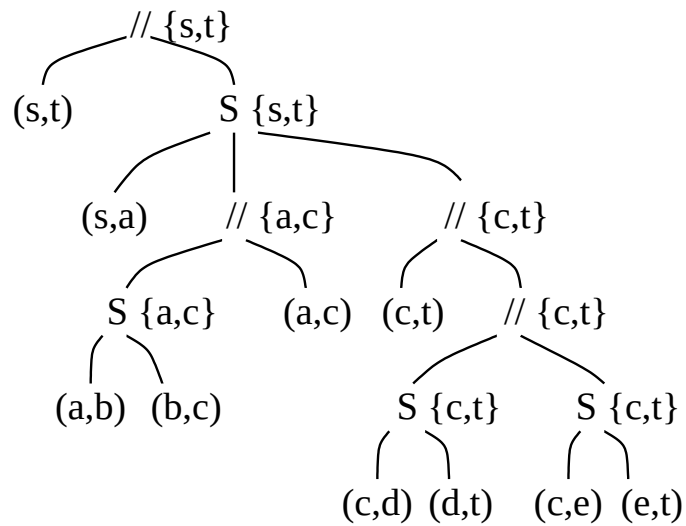
Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition to apply the branching rules and protrusion rule in a bottom-up manner.



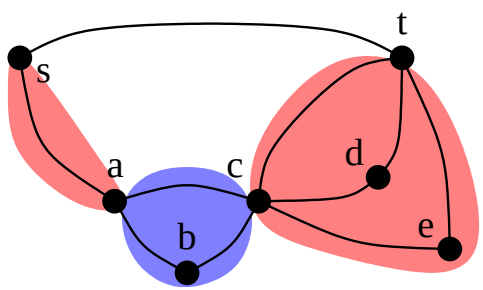


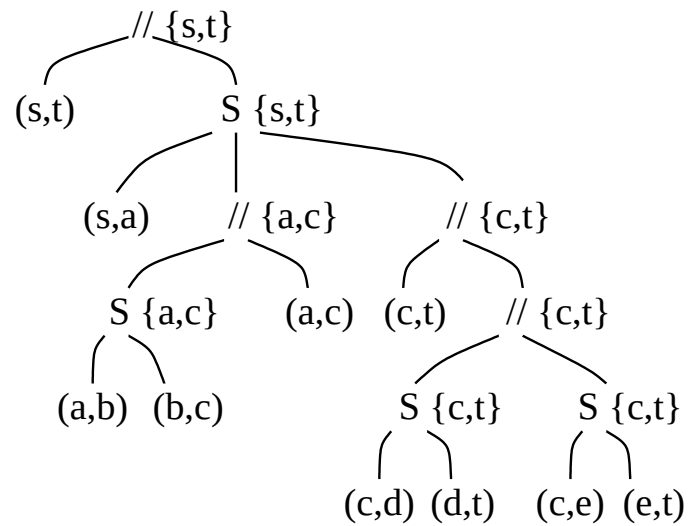
Algorithm outline (1-2)

1. Apply reduction rules and branching rules on every sets $\{x\}$ (with $x \in V \setminus S$)

   Lemma: In a reduced instance, $\boxed{V \setminus S = N_0 \cup N_1 \cup N_2}$

2. Use the extended-SP decomposition to apply the branching rules and protrusion rule in a bottom-up manner.

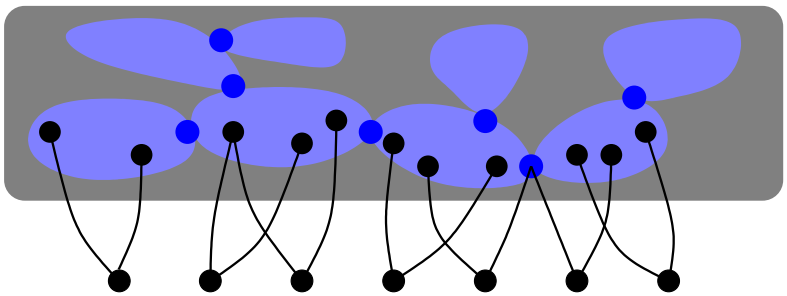




Obs.: $tw(G[V \setminus S] \leqslant 2$ and rules apply on t-protrusions with $t \leqslant 4$

Lemma: The Branch-or-Reduce process leads to $2^{O(k)}$ instances st.
one of them is positive iff the original one was positive

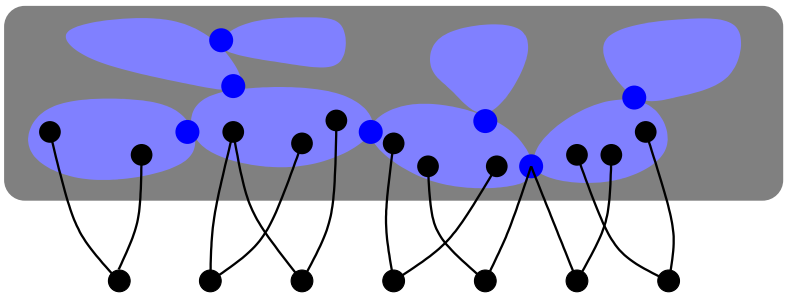
Lemma: The Branch-or-Reduce process leads to $2^{O(k)}$ independent instances st.
one of them is positive iff the original one was positive



$\forall x \in V \setminus S$

- $d(x) = 2$
- $N(x)$ is in some block of $G[S]$
- $G[S \cup \{x\}]$ is K_4-minor free

Lemma: The Branch-or-Reduce process leads to $2^{O(k)}$ independent instances st.
one of them is positive iff the original one was positive

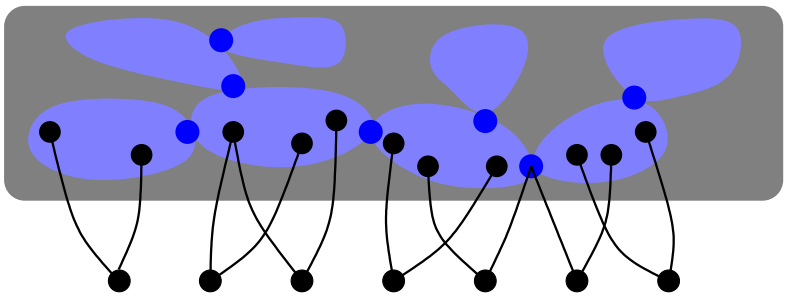


$\forall x \in V \setminus S$

- $d(x) = 2$
- $N(x)$ is in some block of $G[S]$
- $G[S \cup \{x\}]$ is K_4-minor free

Algorithm outline (3)

Lemma: The Branch-or-Reduce process leads to $2^{O(k)}$ independent instances st.
one of them is positive iff the original one was positive



$\forall x \in V \setminus S$

- $d(x) = 2$
- $N(x)$ is in some block of $G[S]$
- $G[S \cup \{x\}]$ is K_4-minor free

Algorithm outline (3)

3. Solve each independent instance
   - Vertex Cover on the auxiliary graph $\tilde{G} = (V \setminus S, \tilde{E})$

$$\boxed{(x, y) \in \tilde{E} \text{ iff } G[G \cup \{x, y\}] \text{ contains a } K_4\text{-minor}}$$

Observation: this is compatible with single exponential time

Skip: $\tilde{G}$ is a circle graph

Algorithm outline (3')

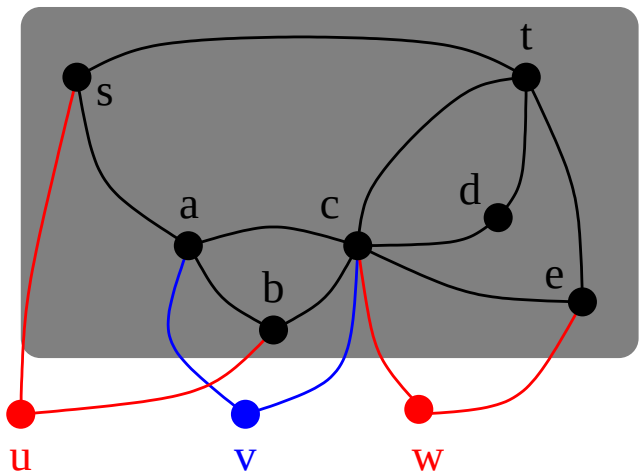
3'. Solve each independent instance in polytime

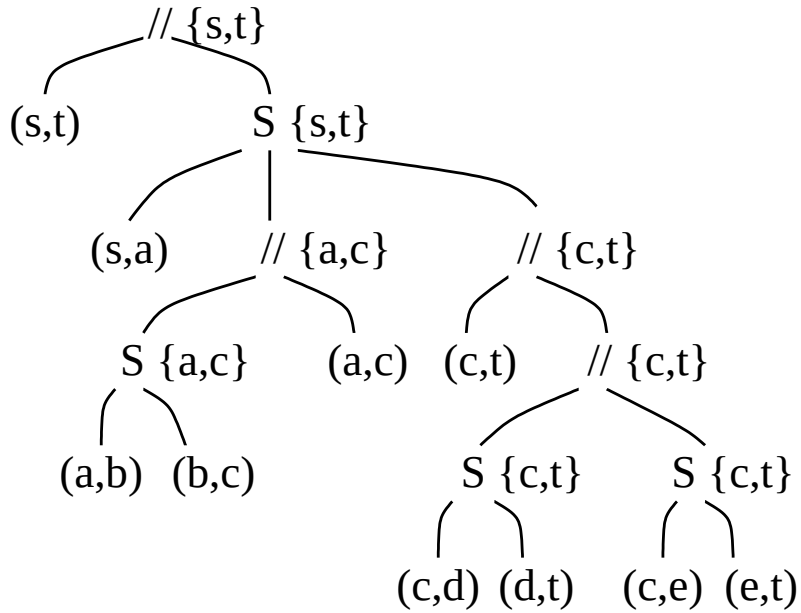
Lemma: The auxiliary graph $\tilde{G}$ is a circle graph

Algorithm outline (3')

3'. Solve each independent instance in polytime

Lemma: The auxiliary graph $\tilde{G}$ is a circle graph



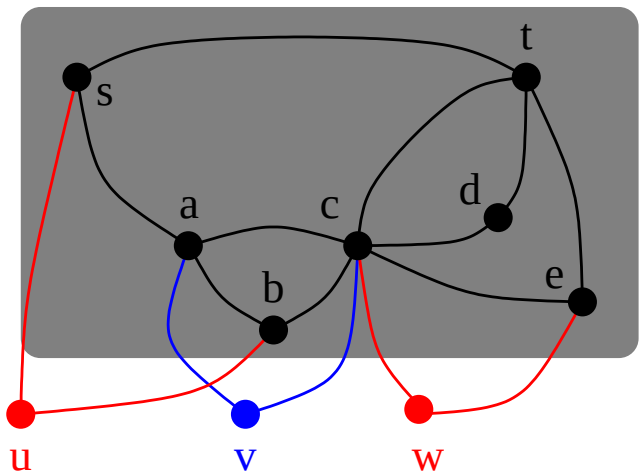


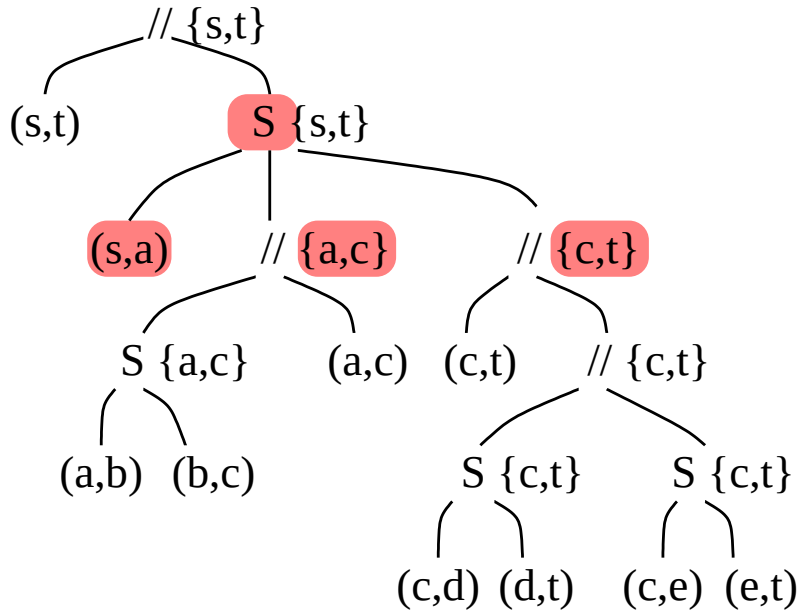
s t

Algorithm outline (3')

3'. Solve each independent instance in polytime

Lemma: The auxiliary graph $\tilde{G}$ is a circle graph



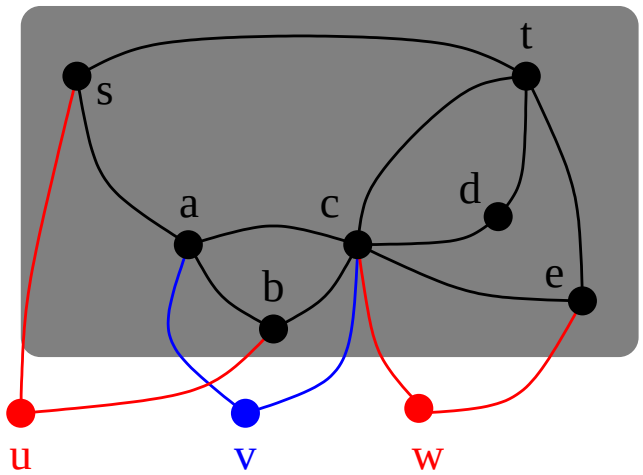


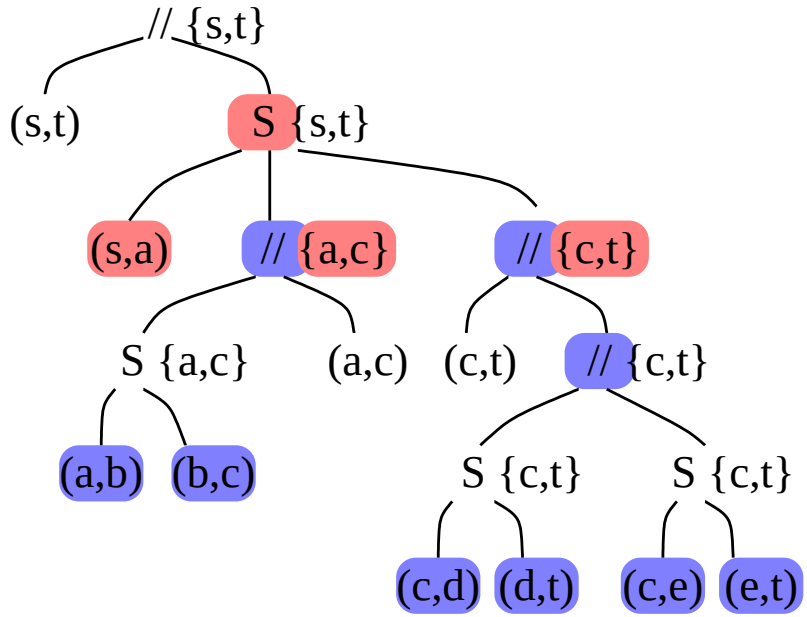
s a c t

Algorithm outline (3')

3'. Solve each independent instance in polytime

Lemma: The auxiliary graph $\tilde{G}$ is a circle graph



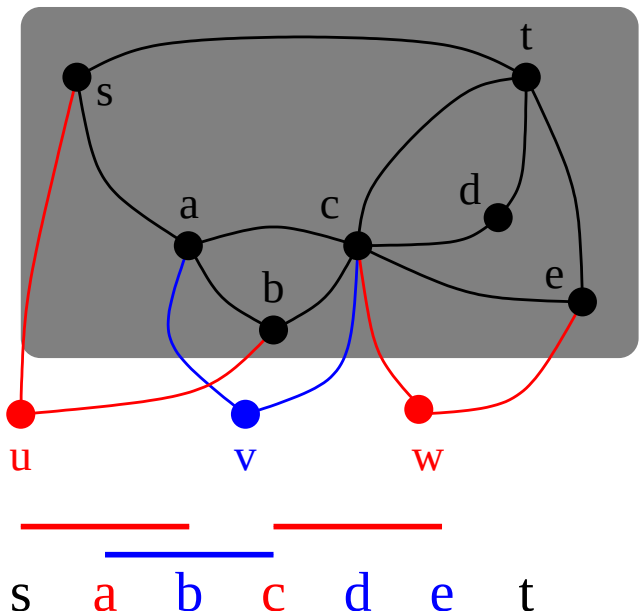


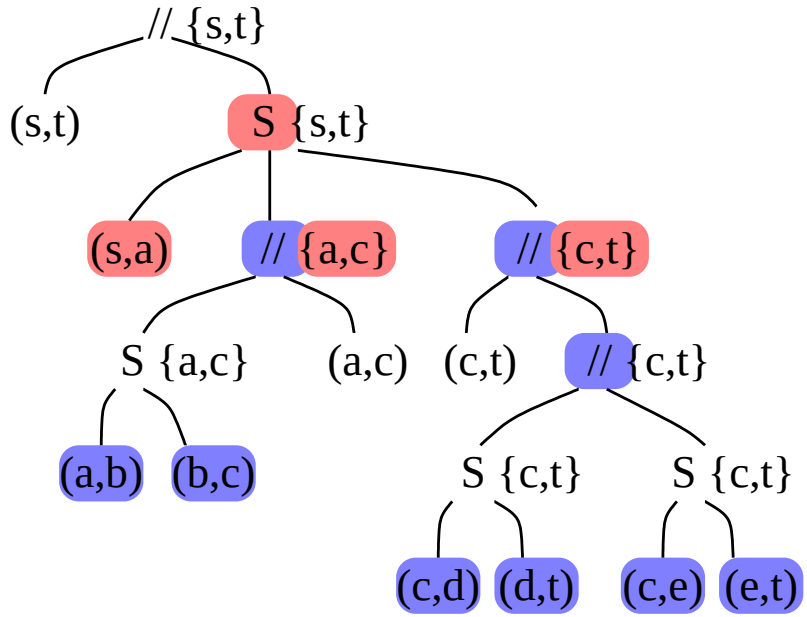
s a b c d e t

Algorithm outline (3')

3'. Solve each independent instance in polytime

Lemma: The auxiliary graph $\tilde{G}$ is a circle graph

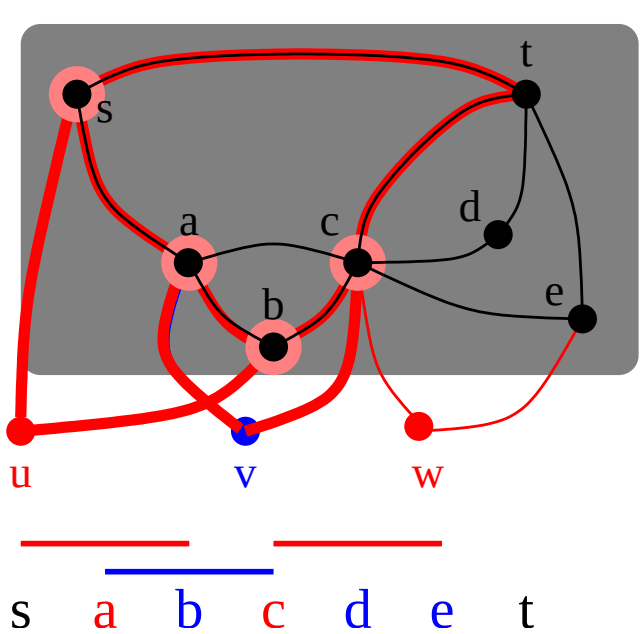


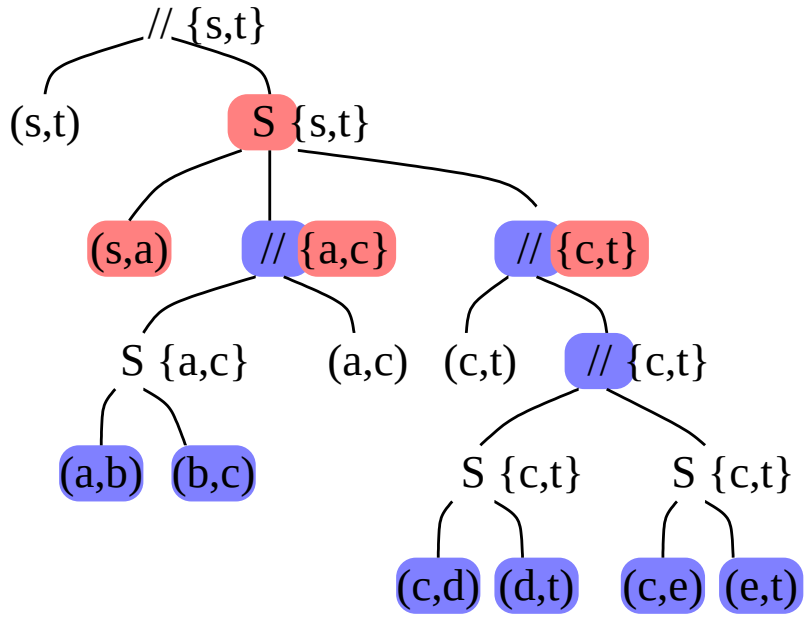


Algorithm outline (3')

3'. Solve each independent instance in polytime

Lemma: The auxiliary graph $\tilde{G}$ is a circle graph





Observation: $G[S \cup \{u, v\}]$ contains a K_4-minor
$\Leftrightarrow$ $I(u)$ and $I(v)$ overlap

Conclusion

Theorem: There exists a single exponential FPT time algorithm for the K_4-Minor Cover problem.

Further works:

Conclusion

Theorem: There exists a single exponential FPT time algorithm for the K_4-Minor Cover problem.

Further works:

1. adpat the branch-or-process rule to address the Outerplanar Graph Vertex Deletion (or equiv. $\{K_{2,3}, K_4\}$-Minor Cover)

Conclusion

Theorem: There exists a single exponential FPT time algorithm for the K_4-Minor Cover problem.

Further works:

1. adpat the branch-or-process rule to address the Outerplanar Graph Vertex Deletion (or equiv. $\{K_{2,3}, K_4\}$-Minor Cover)

2. get rid of the protrusion rule to get an explicit algorithm

3. extend to higher treewidth or to K_t-minor for $t \geqslant 5$