Fixed-parameter tractability of multicut parameterized by the size of the cutset

Dániel Marx

Humboldt-Universität zu Berlin

(Joint work with Igor Razgon)

WorKer 2010: Workshop on Kernelization

Nov 12, 2010

Multiway cut

The classical $s-t$ cut problem:

Given graph G, find a minimum set of edges that separates vertices s and t.

Fact: A minimum $s-t$ cut can be found in polynomial time.

Generalization to more than two terminals:

MULTIWAY CUT	
Input:	A graph G, an integer p, and a set T of terminals
Output:	A set S of at most p edges such that S separates any two vertices of T

Theorem: [Dalhaus et al. 1994] NP-hard already for $|T| = 3$.

Parameterized complexity of MULTIWAY CUT

MULTIWAY CUT can be solved trivially in time $n^{O(p)}$.

Theorem: [M. 2004, Chen et al. 2007] MULTIWAY CUT is fixed-parameter tractable (FPT) parameterized by the size p of the cutset: can be solved in time $O^*(4^p)$.

(Note: the O^* notation hides factors polynomial in the input size.)

MULTICUT

Given pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, a **multicut** is a set of edges that separates s_i and t_i for $i = 1, \ldots, k$.

MULTICUT	
Input:	A graph G, an integer p, pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Output:	A multicut S of size at most p.

MULTICUT

Given pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, a **multicut** is a set of edges that separates s_i and t_i for $i = 1, \ldots, k$.

MULTICUT	
Input:	A graph G, an integer p, pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Output:	A multicut S of size at most p.

Theorem: [M. 2004] MULTICUT can be solved in time $f(k, p) \cdot n^{O(1)}$, i.e., fixed-parameter tractable parameterized by combined parameters k and p.

Theorem: [M. and Razgon 2009] If a solution of size p exists, then we can find a solution of size $2p$ in time $O^*(2^{O(p \log p)})$.

MULTICUT

Given pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, a **multicut** is a set of edges that separates s_i and t_i for $i = 1, \ldots, k$.

MULTICUT	
Input:	A graph G, an integer p, pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Output:	A multicut S of size at most p.

Main result:

MULTICUT can be solved in time $O^*(2^{O(p^3)})$, i.e., fixed-parameter tractable parameterized by p.

Note: Similar result obtained recently by Bousquet, Daligault, and Thomassé (next talk).

Vertex versions

Vertex versions of MULTIWAY CUT and MULTICUT can be analogously defined:

$\Rightarrow$ VERTEX MULTIWAY CUT and VERTEX MULTICUT

Two variants: the separator can contain terminal vertices (unrestricted) or cannot (restricted).

Easy reductions between the two variants and from the edge case to the (restricted) vertex case.

Vertex versions

Vertex versions of MULTIWAY CUT and MULTICUT can be analogously defined:
$\Rightarrow$ VERTEX MULTIWAY CUT and VERTEX MULTICUT

Two variants: the separator can contain terminal vertices (unrestricted) or cannot (restricted).

Easy reductions between the two variants and from the edge case to the (restricted) vertex case.

Same algorithmic result as in the edge case:

Main result:

VERTEX MULTICUT can be solved in time $O^*(2^{O(p^3)})$, i.e., fixed-parameter tractable parameterized by p.

Directed graphs

The problem is much harder and less understood on directed graphs.

New result: (EDGE/VERTEX) DIRECTED MULTICUT is W[1]-hard parameterized by the size p of the cutset.

That is, unlikely that an $f(p) \cdot n^{O(1)}$ time algorithm exists.

Directed graphs

The problem is much harder and less understood on directed graphs.

New result: (EDGE/VERTEX) DIRECTED MULTICUT is W[1]-hard parameterized by the size p of the cutset.

That is, unlikely that an $f(p) \cdot n^{O(1)}$ time algorithm exists.

Several open questions remain:

- What if $k = 2$? $k = 3$?
- Parameterization by both k and p?
- Acyclic graphs?
- DIRECTED MULTIWAY CUT?

Lots of work to be done in this area!

Overview

- Review:
  - important separators
  - algorithm for VERTEX MULTIWAY CUT
- Algorithm for VERTEX MULTICUT:
  - Compression problem.
  - Reduction to ALMOST 2SAT.
  - Creating a nonisolating solution.
  - Reduction to the bipedal case.

Important separators

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $X - Y$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

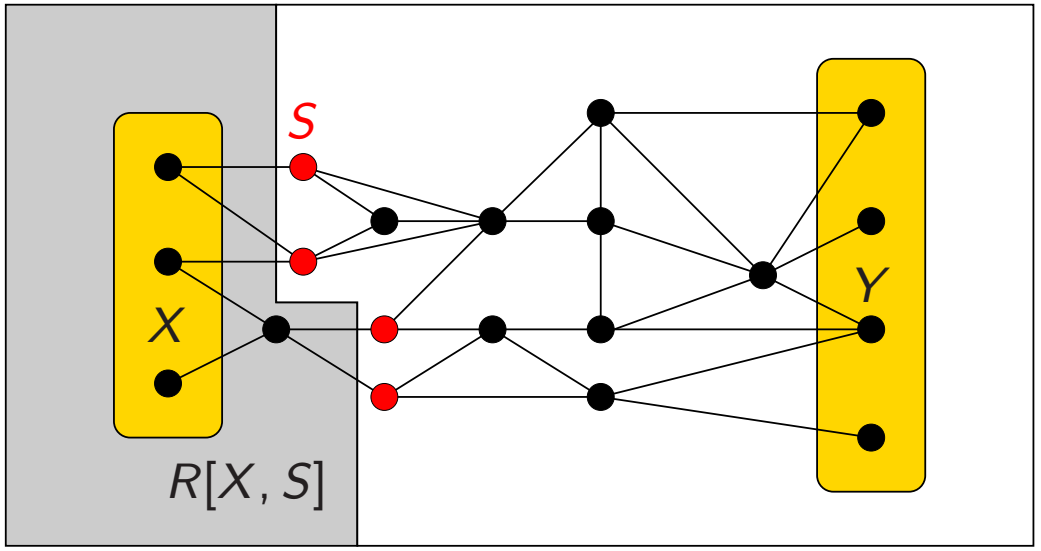
Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.

Important separators

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $X - Y$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.

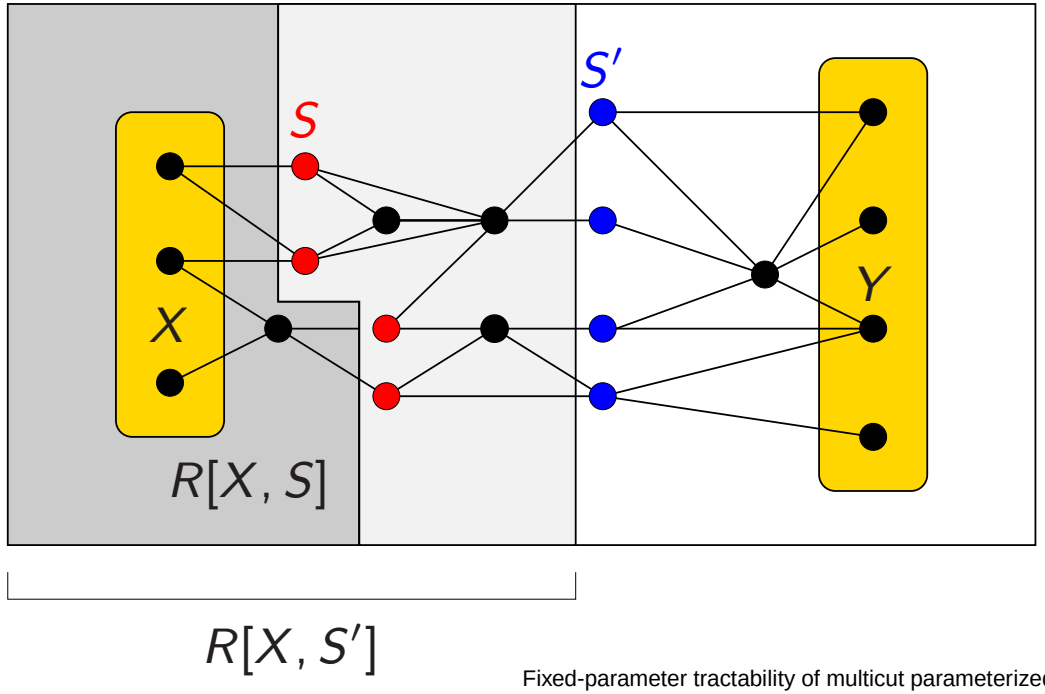


Important separators

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $X - Y$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.

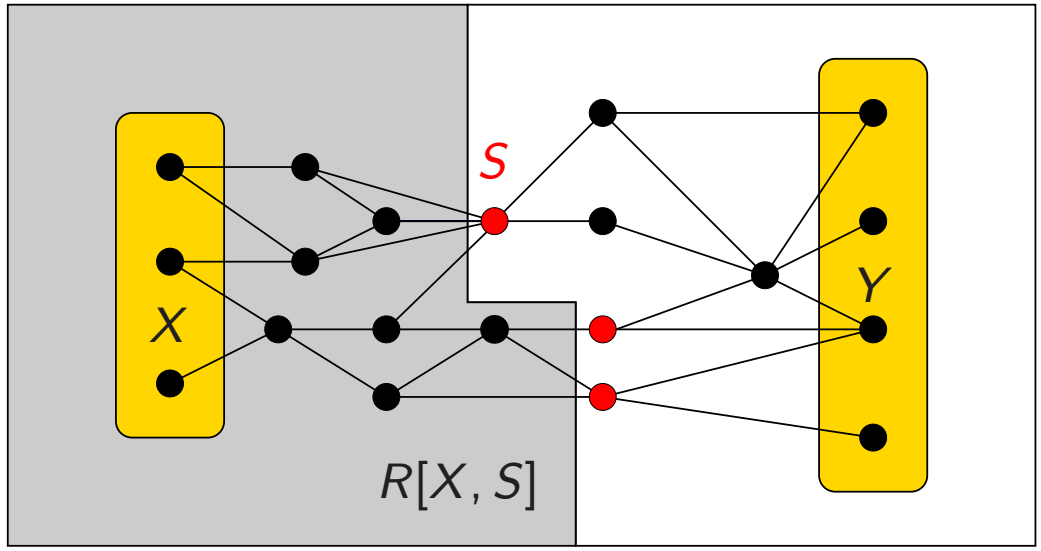


Important separators

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $X - Y$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.

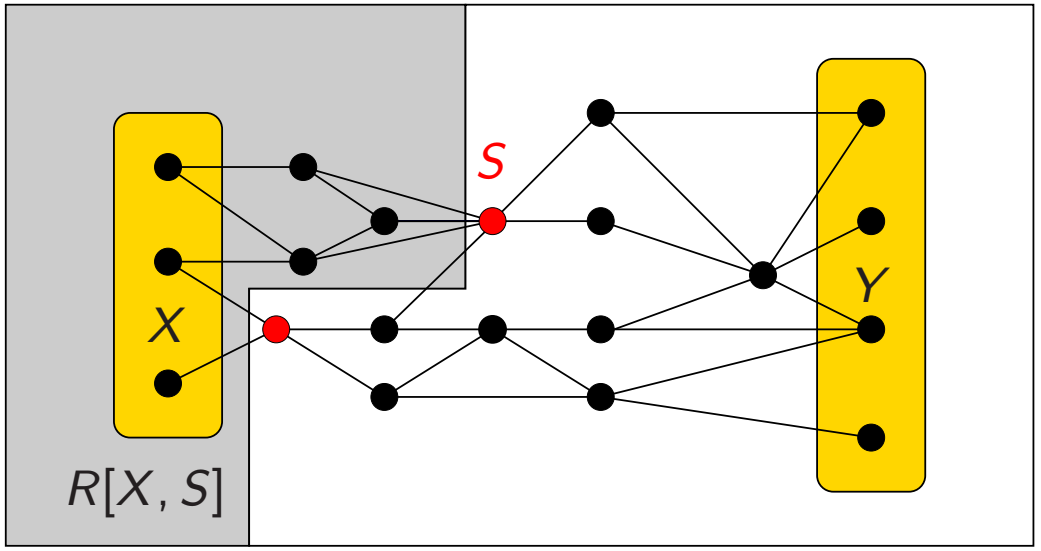


Important separators

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $X - Y$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

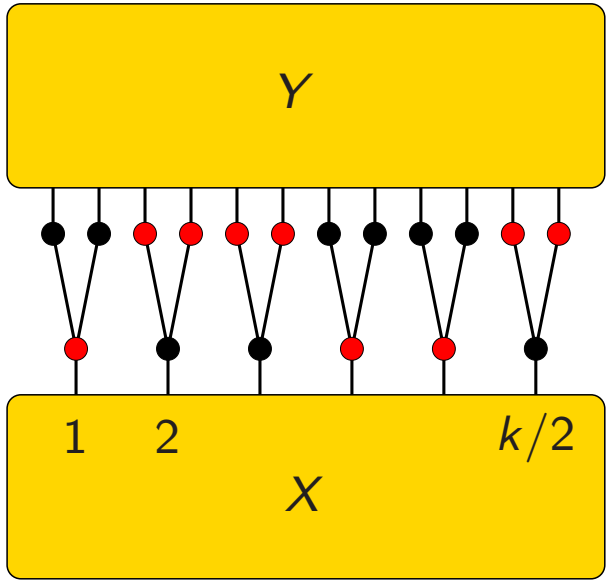
Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.



Important separators

The number of important separators can be exponentially large.

Example:

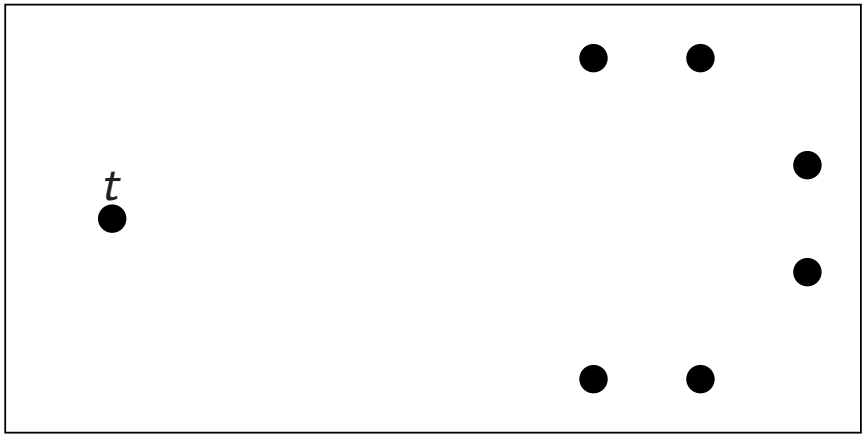


This graph has exactly $2^{k/2}$ important (X, Y)-separators of size at most k.

Theorem: There are at most 4^k important (X, Y)-separators of size at most k.
(Proof is implicit in [Chen, Liu, Lu 2007], worse bound in [M. 2004].)

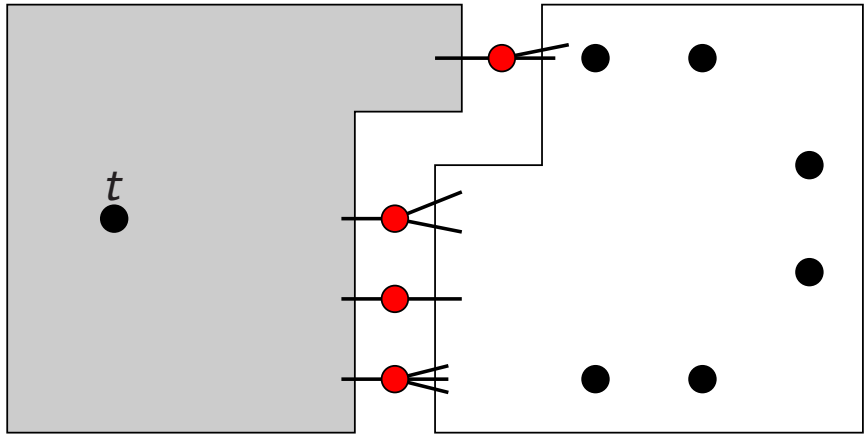
MULTIWAY CUT

Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-separator.



MULTIWAY CUT

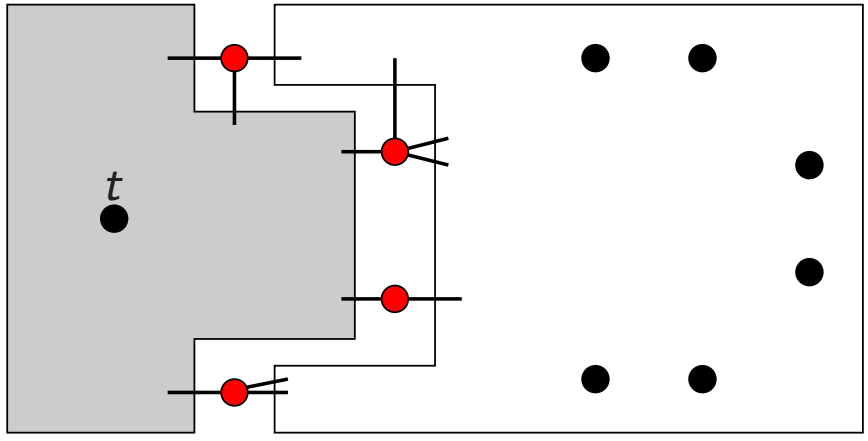
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-separator.



There are many such separators.

MULTIWAY CUT

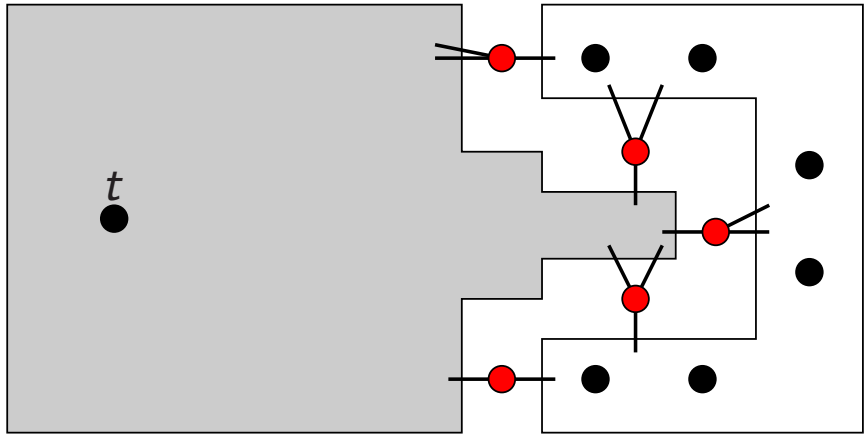
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-separator.



There are many such separators.

MULTIWAY CUT

Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-separator.



There are many such separators.

But a separator farther from t and closer to $T \setminus t$ seems to be more useful!

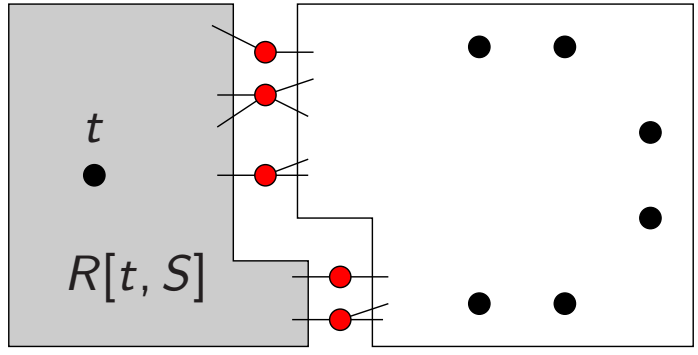
MULTIWAY CUT *and important separators*

Pushing Lemma: Let $t \in T$. The MULTIWAY CUT problem has a solution that contains an important $(t, T \setminus t)$-separator.

MULTIWAY CUT *and important separators*

Pushing Lemma: Let $t \in T$. The MULTIWAY CUT problem has a solution that contains an important $(t, T \setminus t)$-separator.

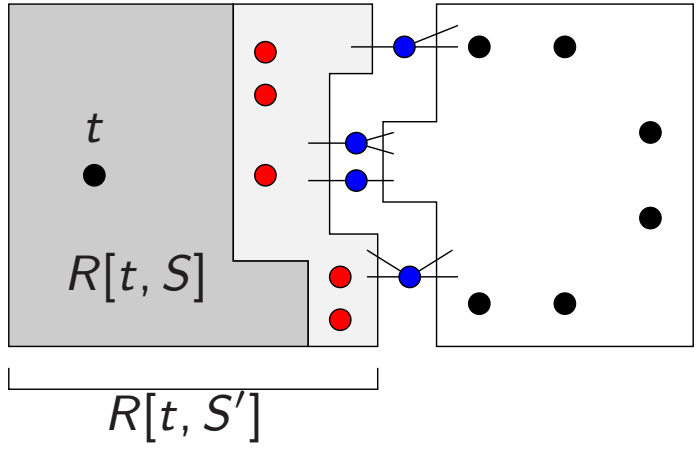
Proof: Let Q be a solution and let $S \subseteq Q$ be the vertices reachable from t.



MULTIWAY CUT *and important separators*

Pushing Lemma: Let $t \in T$. The MULTIWAY CUT problem has a solution that contains an important $(t, T \setminus t)$-separator.

Proof: Let Q be a solution and let $S \subseteq Q$ be the vertices reachable from t.

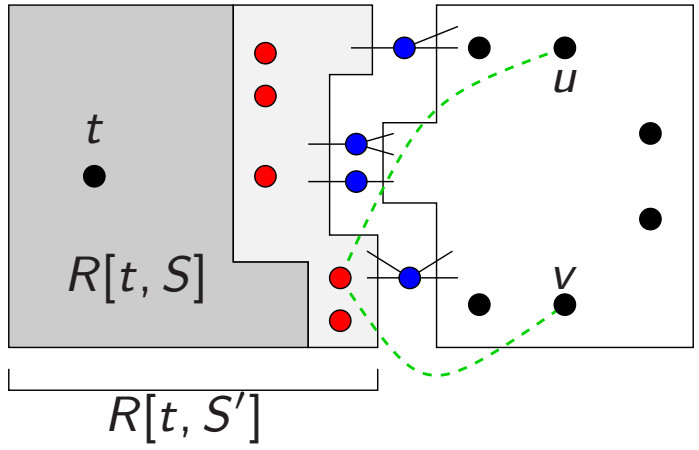


If S is not important, then there is an important S' with $R[t, S] \subset R[t, S']$ and $|S'| \leq |S|$. Replace Q with $Q^* := (Q \setminus S) \cup S' \Rightarrow |Q^*| \leq |Q|$

MULTIWAY CUT *and important separators*

Pushing Lemma: Let $t \in T$. The MULTIWAY CUT problem has a solution that contains an important $(t, T \setminus t)$-separator.

Proof: Let Q be a solution and let $S \subseteq Q$ be the vertices reachable from t.



If S is not important, then there is an important S' with $R[t, S] \subset R[t, S']$ and $|S'| \leq |S|$. Replace Q with $Q^* := (Q \setminus S) \cup S' \Rightarrow |Q^*| \leq |Q|$

Q^* is a multiway cut: (1) There is no t-u path in $G \setminus Q^*$ and (2) a u-v path in $G \setminus Q^*$ must go through S, but S' separates S from u, contradiction.

Algorithm for VERTEX MULTIWAY CUT

1. If every vertex of T is in a different component, then we are done.
2. Let $t \in T$ be a vertex that is not separated from every $T \setminus t$.
3. Branch on a choice of an important $(t, T \setminus t)$ separator S of size at most p.
4. Set $G := G \setminus S$ and $p := p - |S|$.
5. Go to step 1.

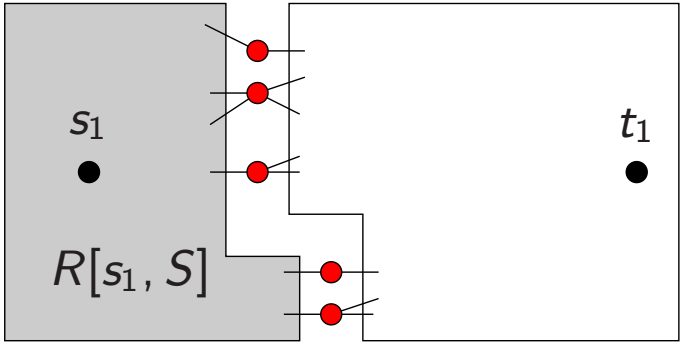
We branch into at most 4^p directions at most p times (better analysis shows that the size of search tree is at most 4^p).

Multicut

Does this approach work for MULTICUT?

We know that s_1 is separated from t_1, but we do not know which vertices of s_2, t_2, ..., s_k, t_k are separated from t_1.

The solution contains an $s_1 - t_1$ separator S, but replacing it with an important $s_1 - t_1$ separator S' can create an s_i-t_i path.

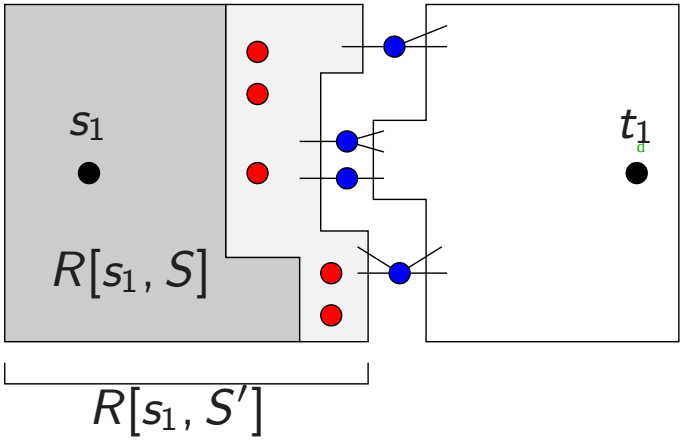


Multicut

Does this approach work for MULTICUT?

We know that s_1 is separated from t_1, but we do not know which vertices of s_2, t_2, ..., s_k, t_k are separated from t_1.

The solution contains an $s_1 - t_1$ separator S, but replacing it with an important $s_1 - t_1$ separator S' can create an s_i-t_i path.

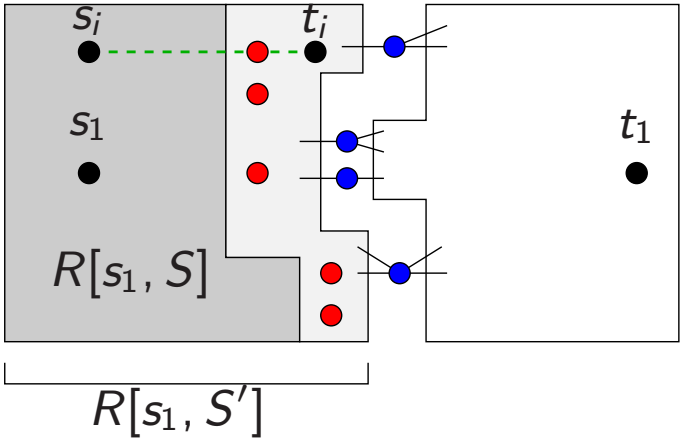


Multicut

Does this approach work for MULTICUT?

We know that s_1 is separated from t_1, but we do not know which vertices of s_2, t_2, ..., s_k, t_k are separated from t_1.

The solution contains an $s_1 - t_1$ separator S, but replacing it with an important $s_1 - t_1$ separator S' can create an s_i-t_i path.



The compression problem

A standard technique in the design of parameterized algorithms: solve the compression problem first.

MULTICUT COMPRESSION

Input:	A graph G, an integer p, pairs of vertices (s_1, t_1), ..., (s_k, t_k), and a multicut W.
Output:	A multicut S of size at most p.

Our first goal:

Lemma: MULTICUT COMPRESSION is FPT parameterized by p and $|W|$.

Using the compression problem

Lemma: MULTICUT COMPRESSION is FPT parameterized by p and $|W|$.

Two ways of using this:

- **Method 1:** The polynomial-time approximation algorithm of [Gupta 2003] finds a solution of size OPT^2 in polynomial time: we get a solution W with $|W| \leq p^2$.
- **Method 2:** Use iterative compression. We can reduce VERTEX MULTICUT to $|V(G)|$ calls of MULTICUT COMPRESSION with $|W| = p + 1$.

Using the compression problem

Lemma: MULTICUT COMPRESSION is FPT parameterized by p and $|W|$.

Two ways of using this:

- **Method 1:** The polynomial-time approximation algorithm of [Gupta 2003] finds a solution of size OPT^2 in polynomial time: we get a solution W with $|W| \leq p^2$.
- **Method 2:** Use iterative compression. We can reduce VERTEX MULTICUT to $|V(G)|$ calls of MULTICUT COMPRESSION with $|W| = p + 1$.

We can solve MULTICUT COMPRESSION in time $O^*(2^{O((p+\log|W|)^3+|W|\log|W|)})$

$\Rightarrow$ We can solve VERTEX MULTICUT in time $O^*(2^{O(p^3)})$.

The compression problem

MULTICUT COMPRESSION*

Input: A graph G, an integer p, pairs of vertices (s_1, t_1), ..., (s_k, t_k), and a multicut W.

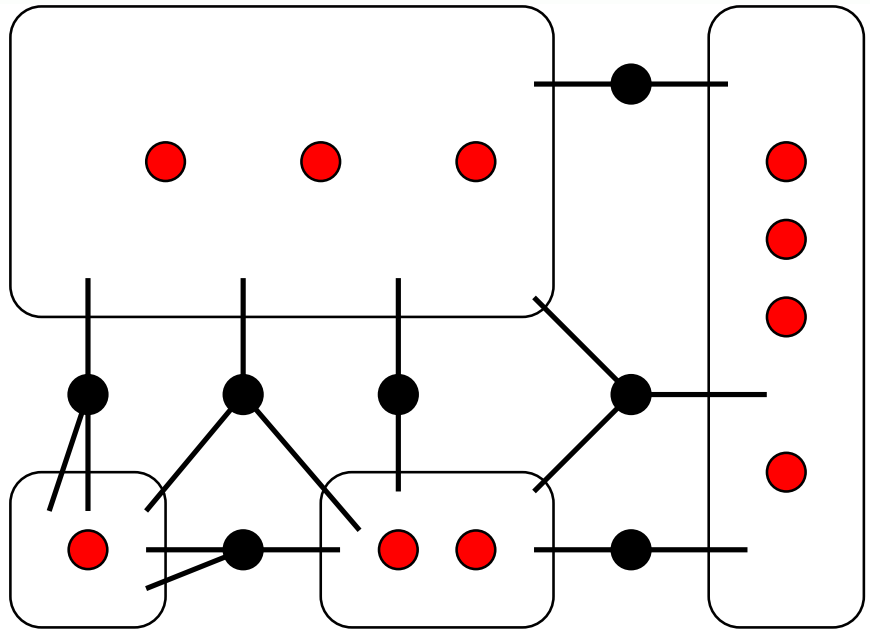
Output: A multicut S of size at most p such that (1) $S \cap W = \emptyset$ and (2) S is a multiway cut of W.

Easy reduction from the original MULTICUT COMPRESSION to this MULTICUT COMPRESSION*:

- To ensure (1), we guess the intersection $S \cap W$ and remove it from G.
- To ensure (2), we guess the way the components of $G \setminus S$ partition W, and contract each class into a single vertex.

In the rest of talk, we show that MULTICUT COMPRESSION* is FPT parameterized by p and $|W|$.

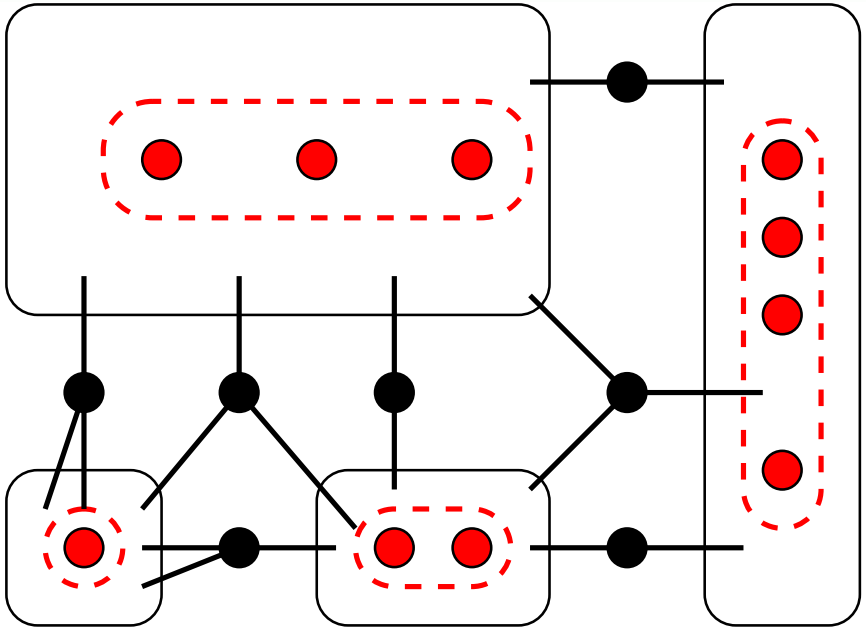
Guessing a partition



We guess the way the solution partitions W and contract each class into a single vertex (at most $|W|^{|W|}$ possibilities).

The contraction does not make the problem any easier, and when we guess the correct partition, then it does not make it any harder.

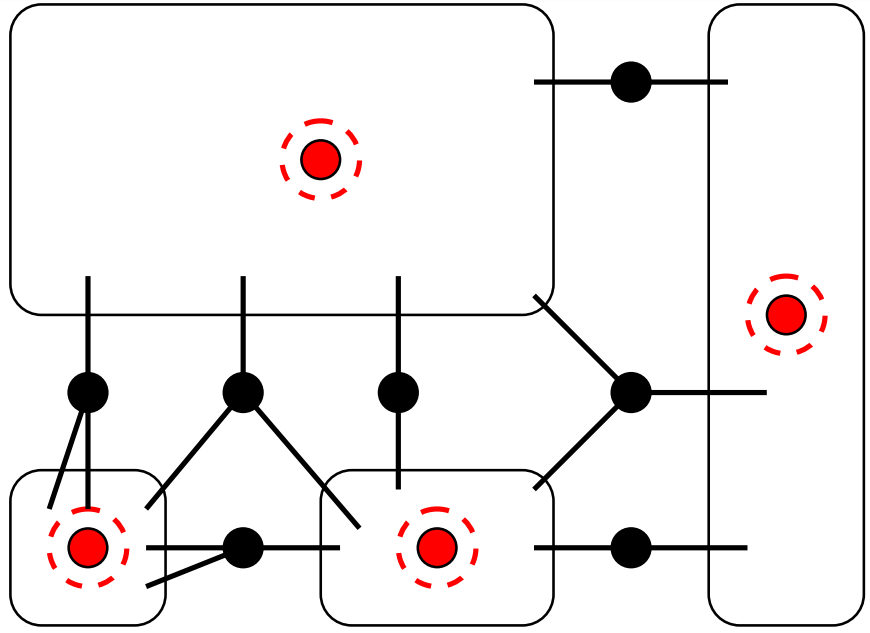
Guessing a partition



We guess the way the solution partitions W and contract each class into a single vertex (at most $|W|^{|W|}$ possibilities).

The contraction does not make the problem any easier, and when we guess the correct partition, then it does not make it any harder.

Guessing a partition

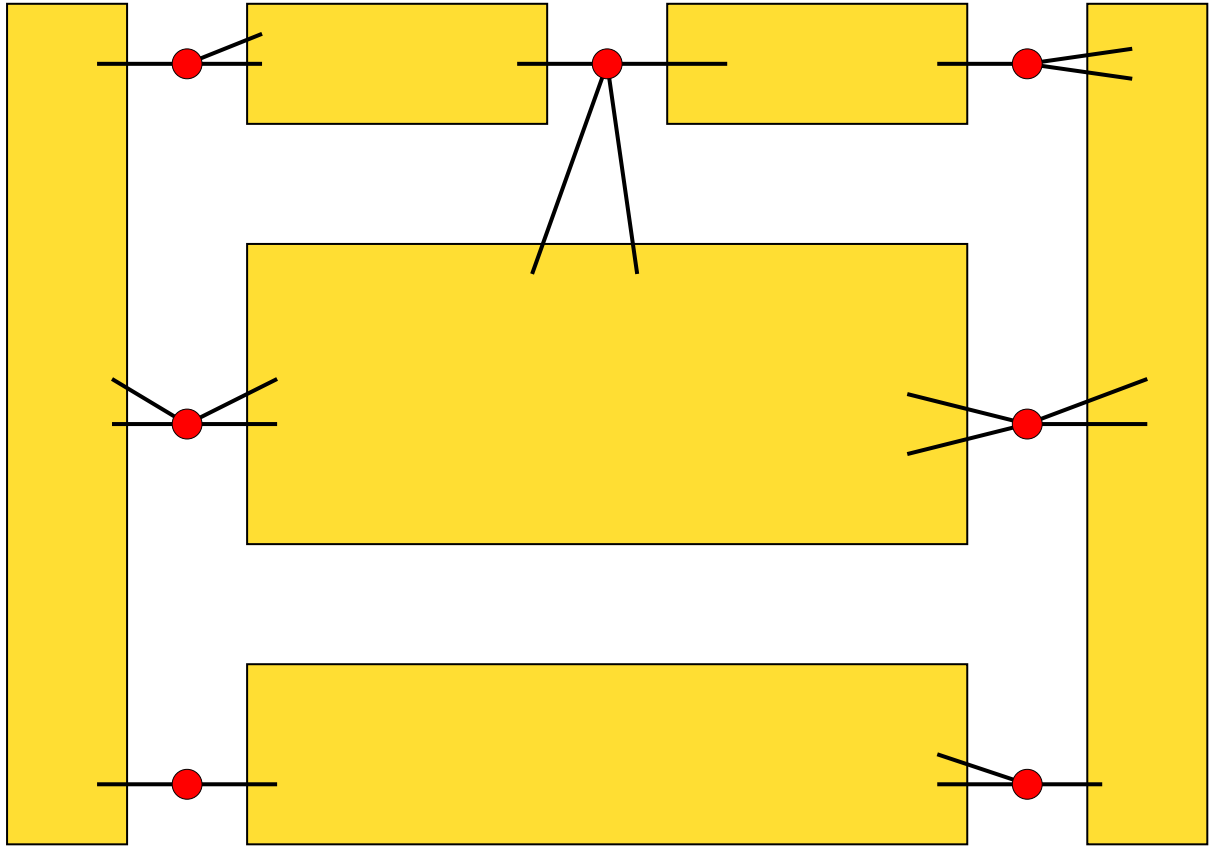


We guess the way the solution partitions W and contract each class into a single vertex (at most $|W|^{|W|}$ possibilities).

The contraction does not make the problem any easier, and when we guess the correct partition, then it does not make it any harder.

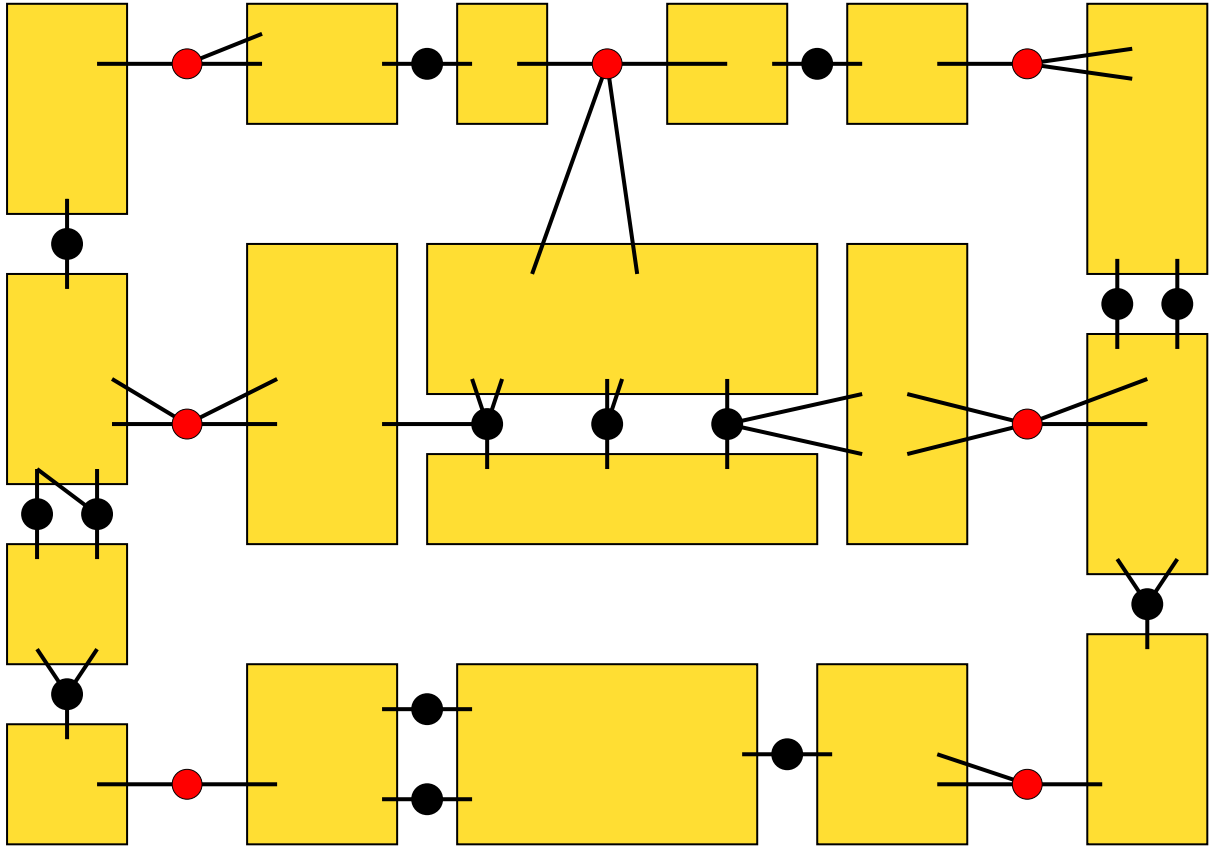
MULTICUT COMPRESSION*

An instance looks like this (the red vertices are in W):



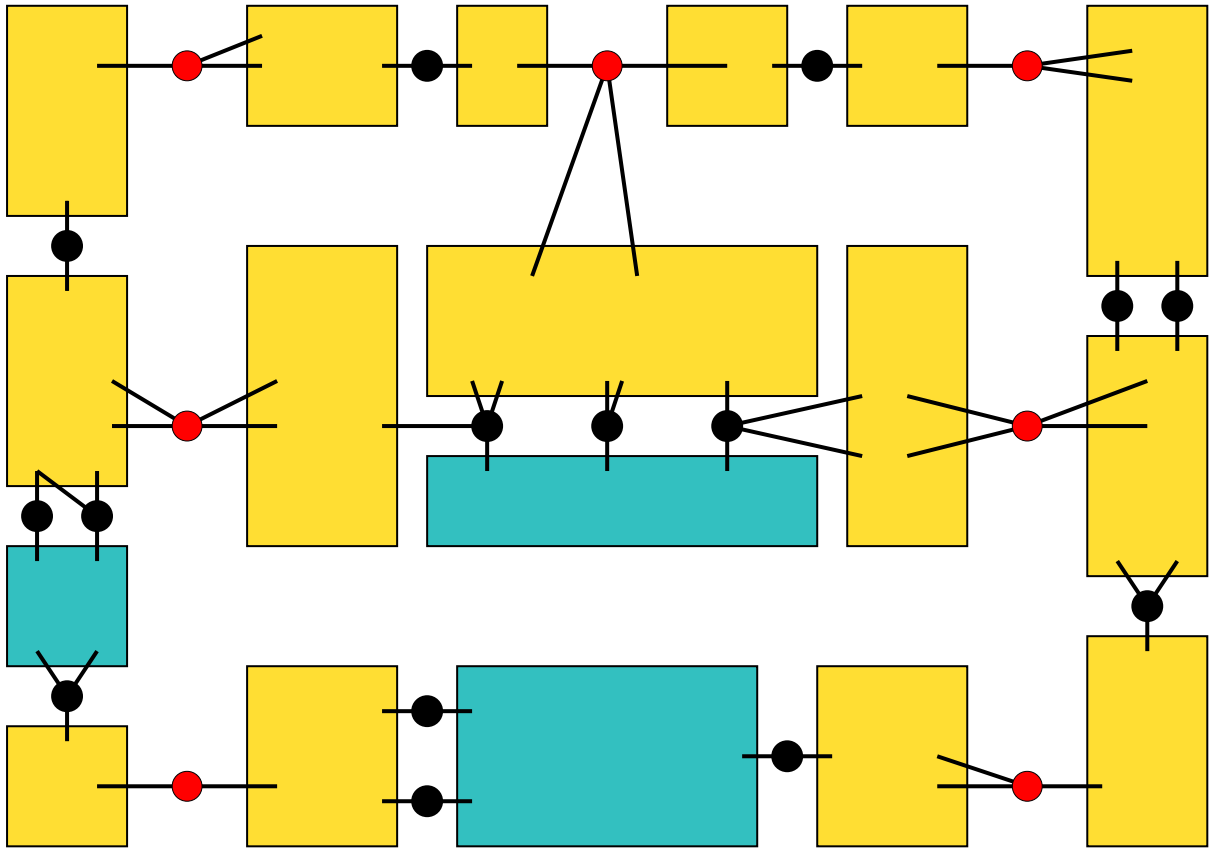
MULTICUT COMPRESSION*

An instance looks like this (the red vertices are in W):



MULTICUT COMPRESSION*

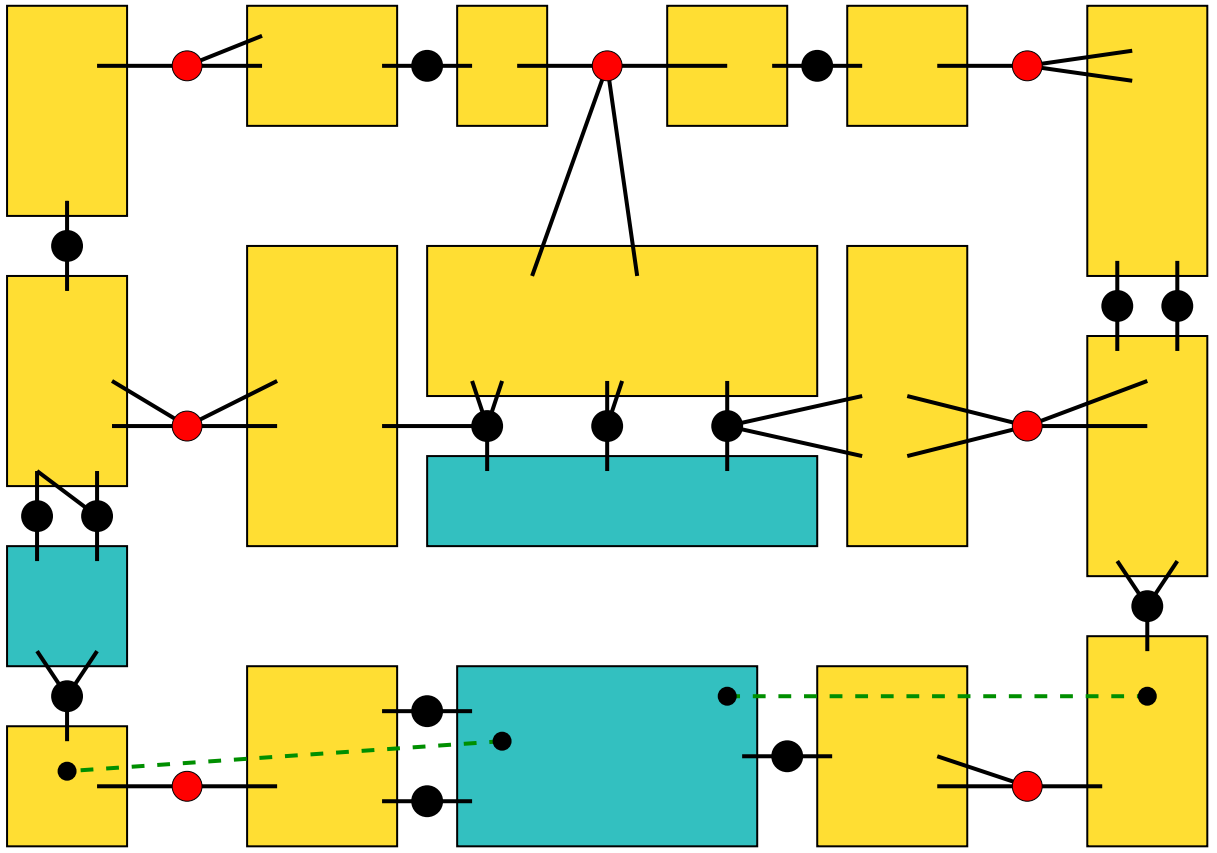
An instance looks like this (the red vertices are in W):



Isolated part: vertices of $G \setminus W$ separated from W by the solution.

MULTICUT COMPRESSION*

An instance looks like this (the red vertices are in W):



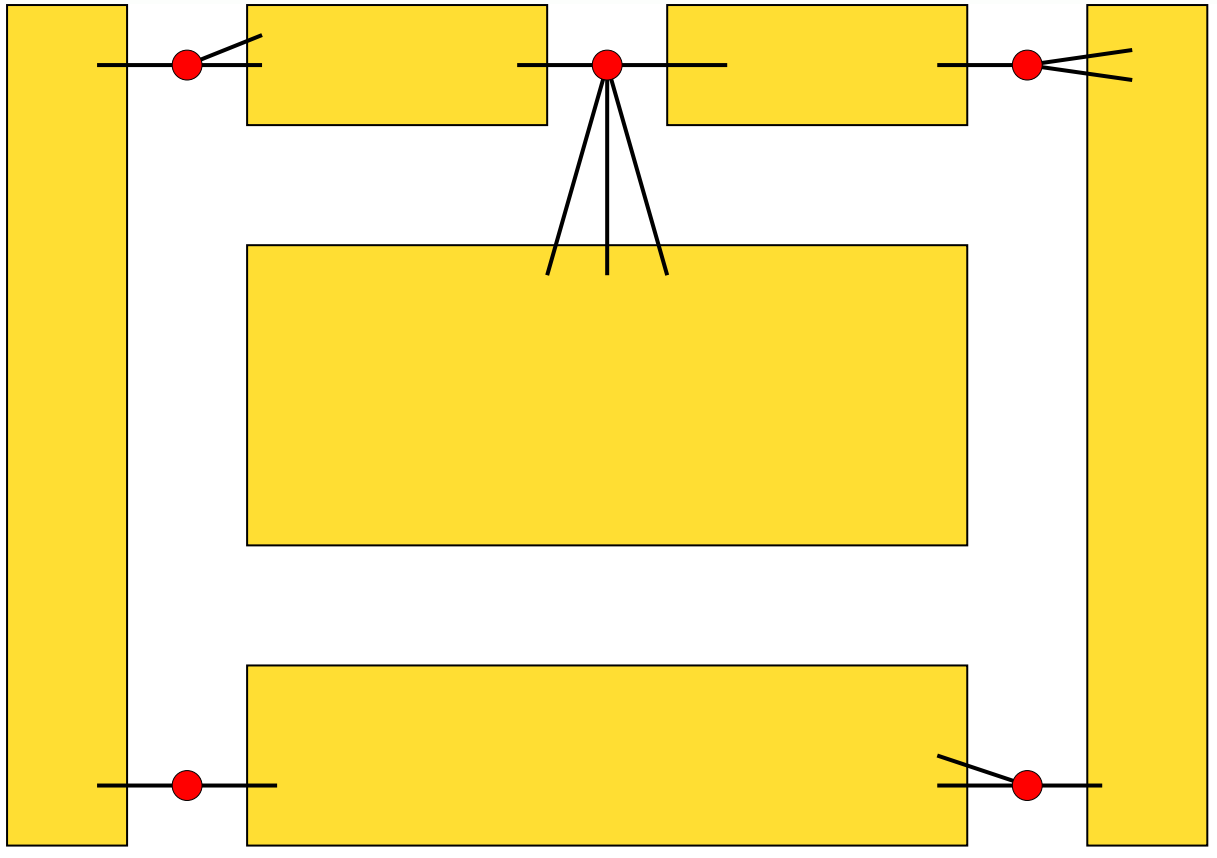
Isolated part: vertices of $G \setminus W$ separated from W by the solution.

A special case

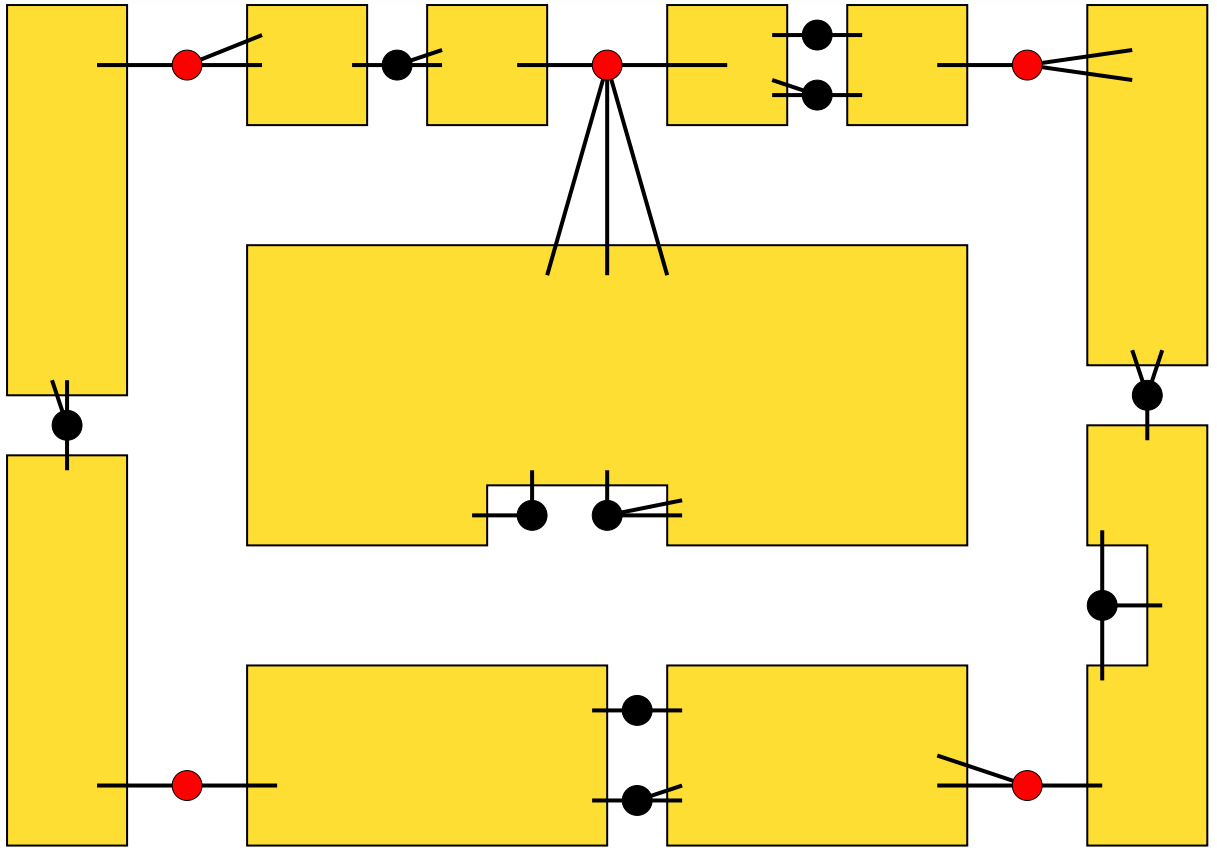
We can solve MULTICUT COMPRESSION* by reduction to ALMOST 2SAT if the following two conditions hold:

- (1) There is a solution where the isolated part is empty (“nonisolating solution”).
- (2) Every component of $G \setminus W$ has at most two legs, i.e, adjacent to at most two vertices of W (“bipedal instance”).

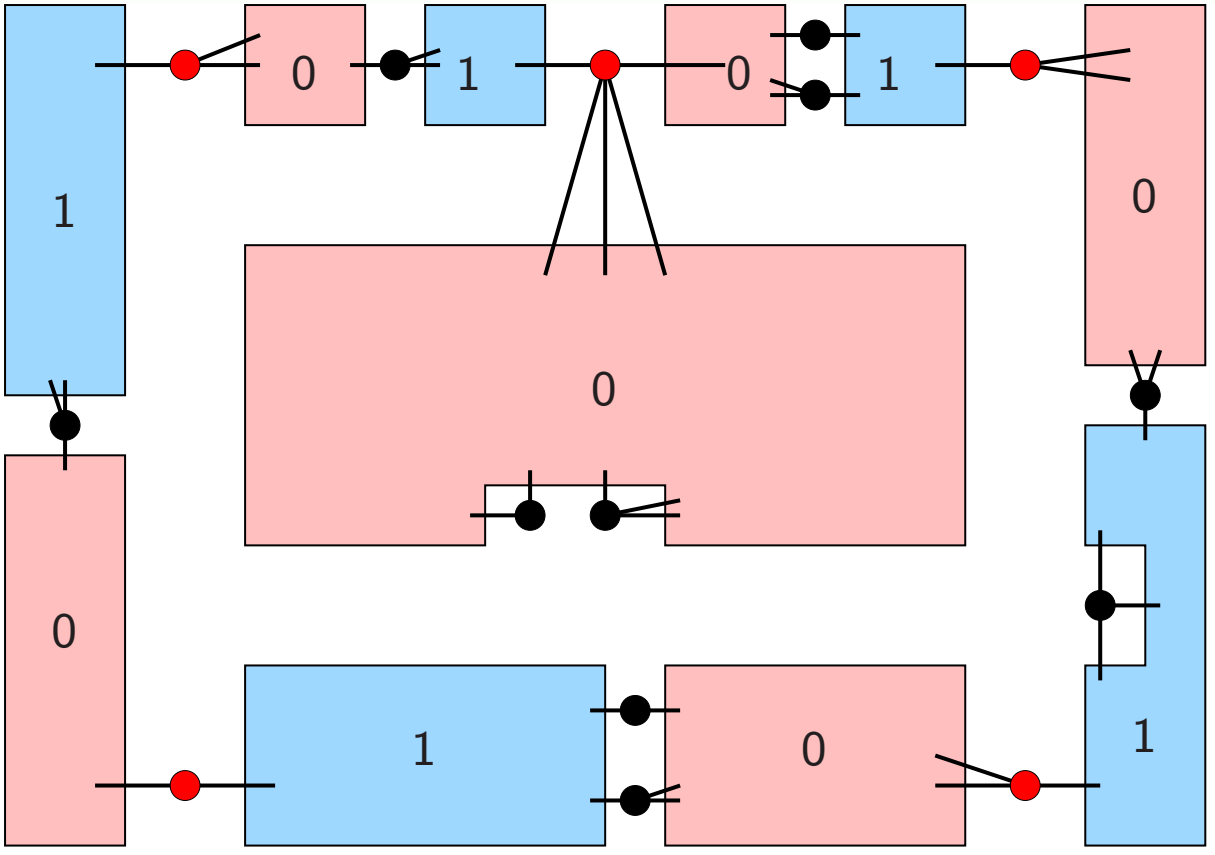
Special case of MULTICUT COMPRESSION*



Special case of MULTICUT COMPRESSION*



Special case of MULTICUT COMPRESSION*



Each vertex is either deleted, reachable from leg 0, or reachable from leg 1.

Almost 2SAT

A 2SAT formula is a conjunction of 2-clauses, e.g.,

$$(x_1 \vee \bar{x}_3) \wedge (x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_4).$$

Fact: A satisfying assignment for a satisfiable 2SAT formula can be found in linear time.

Fact: It is NP-hard to find an assignment that satisfies the maximum number of clauses of a 2SAT formula.

Almost 2SAT

A 2SAT formula is a conjunction of 2-clauses, e.g.,

$$(x_1 \vee \bar{x}_3) \wedge (x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_4).$$

Fact: A satisfying assignment for a satisfiable 2SAT formula can be found in linear time.

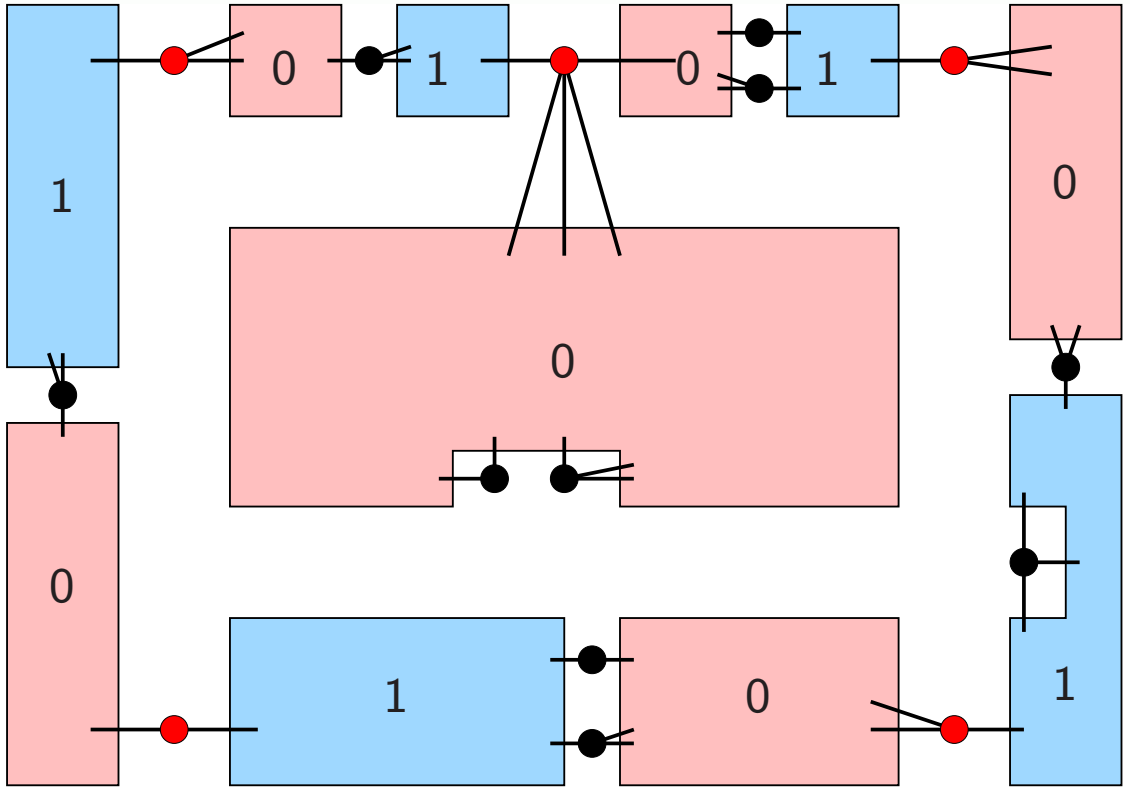
Fact: It is NP-hard to find an assignment that satisfies the maximum number of clauses of a 2SAT formula.

Theorem: [O'Sullivan and Razgon 2008] In time $O^*(15^k)$, we can decide if a 2SAT formula can be made satisfiable by the deletion of k clauses.

Easy consequence (exercise):

Theorem: In time $O^*(15^k)$, we can decide if a 2SAT formula can be made satisfiable by the deletion of k variables.

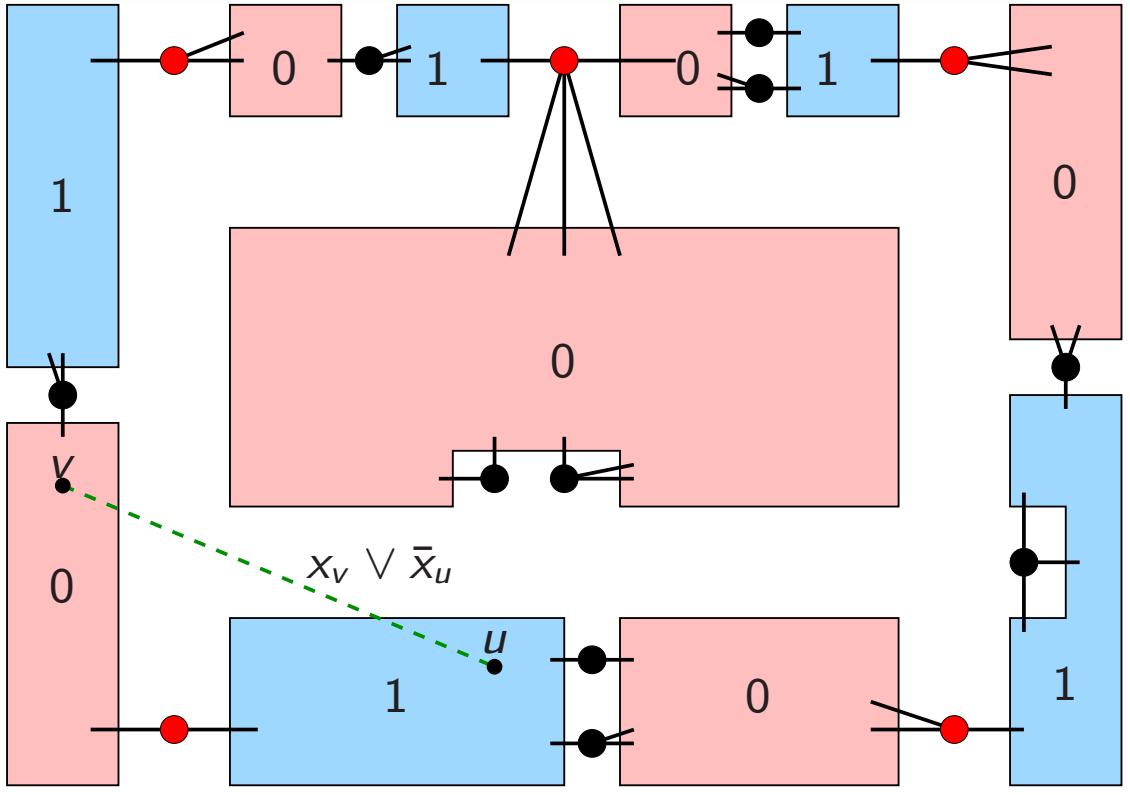
Reduction to ALMOST 2SAT



Each vertex v of $G \setminus W$ is represented by a variable x_v:

$$x_v = 0 \iff v \text{ is reachable from leg 0}$$
$$x_v = 1 \iff v \text{ is reachable from leg 1}$$
$$x_v \text{ is deleted} \iff v \text{ is deleted}$$

Reduction to ALMOST 2SAT



Each vertex v of $G \setminus W$ is represented by a variable x_v:

$$x_v = 0 \iff v \text{ is reachable from leg 0}$$
$$x_v = 1 \iff v \text{ is reachable from leg 1}$$
$$x_v \text{ is deleted} \iff v \text{ is deleted}$$

Reduction to ALMOST 2SAT

We introduce 4 groups of clauses:

- Group 1: $(x_u \rightarrow x_v)$, $(x_v \rightarrow x_u)$ for every adjacent $u, v \in V(G) \setminus W$.
- Group 2: If u is a neighbor of leg $b \in \{0, 1\}$ of the component, then $(x_u = b)$.
- Group 3: If $s_i, t_i \notin W$, and leg b_s of (the component of) s_i is the same as leg b_t of s_i, then $(x_{s_i} \neq b_s \vee x_{t_i} \neq b_t)$.
- Group 4: If $s_i \in W$, $t_i \notin W$, and s_i is leg b of t_i, the $(x_{t_i} \neq b)$.

Lemma:

(1) If there is a nonisolating solution S of size p, then deleting the variables corresponding to S makes these clauses satisfiable.

(2) If deleting a set S of variables makes the clauses satisfiable, then the set of vertices corresponding to S is a solution.

A special case

We have seen that MULTICUT COMPRESSION* can be solved in time $O^*(15^p)$ by reduction to ALMOST 2SAT if the following two conditions hold:

- (1) There is a solution where the isolated part is empty (“nonisolating solution”).
- (2) Every component of $G \setminus W$ has at most two legs, i.e, adjacent to at most two vertices of W (“bipedal instance”).

A special case

We have seen that MULTICUT COMPRESSION* can be solved in time $O^*(15^p)$ by reduction to ALMOST 2SAT if the following two conditions hold:

- (1) There is a solution where the isolated part is empty (“nonisolating solution”).
- (2) Every component of $G \setminus W$ has at most two legs, i.e, adjacent to at most two vertices of W (“bipedal instance”).

Next we show how to ensure that condidition (1) holds.

Intuitively, we want to cut away the isolated part (but we don’t know where it is).

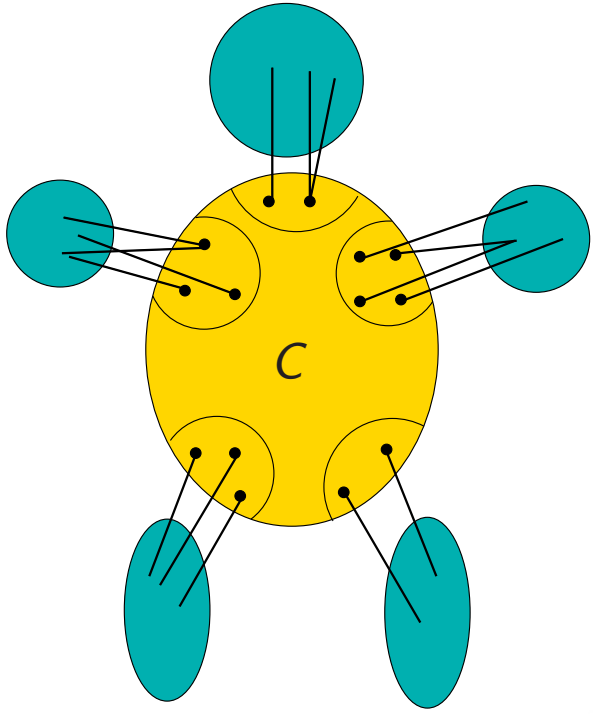
Most interesting part of the algorithm!

Torso

We use the following operation to cut away the isolated part.

Definition: For a set C of vertices of G, graph torso(G, C) has vertex set C and $a, b \in C$ are adjacent iff they are adjacent in G or there is an $a - b$ path internally disjoint from C.

In other words: for each component K of $G \setminus C$, we add a clique where K is attached to C.

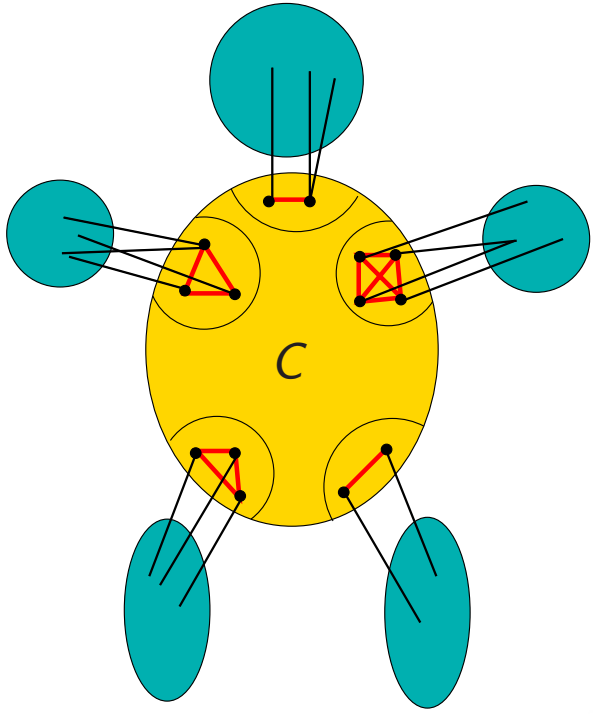


Torso

We use the following operation to cut away the isolated part.

Definition: For a set C of vertices of G, graph torso(G, C) has vertex set C and $a, b \in C$ are adjacent iff they are adjacent in G or there is an $a - b$ path internally disjoint from C.

In other words: for each component K of $G \setminus C$, we add a clique where K is attached to C.

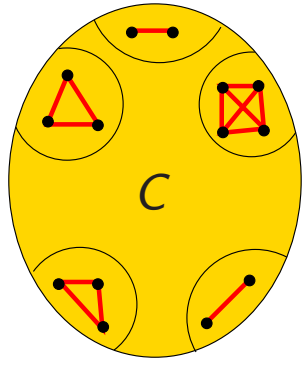


Torso

We use the following operation to cut away the isolated part.

Definition: For a set C of vertices of G, graph torso(G, C) has vertex set C and $a, b \in C$ are adjacent iff they are adjacent in G or there is an $a - b$ path internally disjoint from C.

In other words: for each component K of $G \setminus C$,
we add a clique where K is attached to C.



Torso

We use the following operation to cut away the isolated part.

Definition: For a set C of vertices of G, graph torso(G, C) has vertex set C and $a, b \in C$ are adjacent iff they are adjacent in G or there is an $a - b$ path internally disjoint from C.

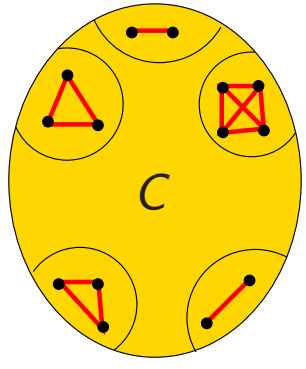
In other words: for each component K of $G \setminus C$,
we add a clique where K is attached to C.

Fact: If $s, t \in C$, and $S \subseteq C$, then

$$S \text{ is an } s - t \text{ separator in } G$$

$$\Updownarrow$$

$$S \text{ is an } s - t \text{ separator in torso}(G, C).$$



Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

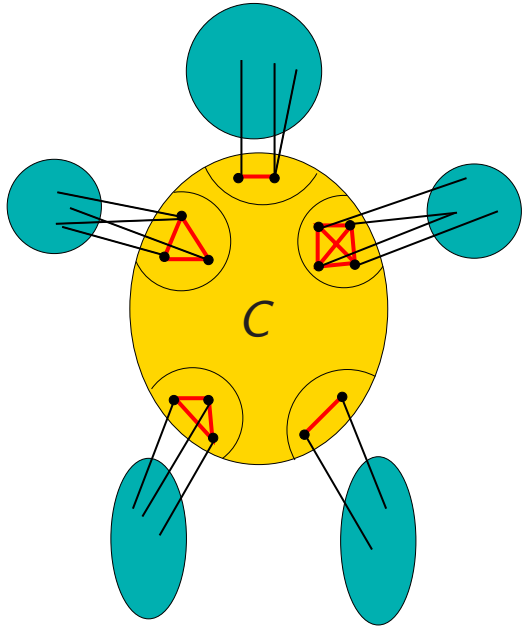
If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

The pairs (s_i, t_i) need to be changed if s_i or t_i is in Z.

For $v \in Z$, let $K(v)$ be the corresponding clique.
For $v \notin Z$, let $K(v) = \{v\}$.

We replace every pair (s_i, t_i) with the pairs $K(s_i) \times K(t_i)$.



Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

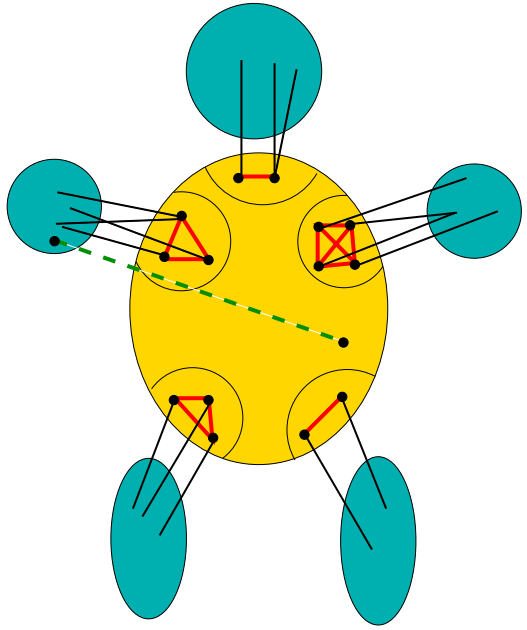
If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

The pairs (s_i, t_i) need to be changed if s_i or t_i is in Z.

For $v \in Z$, let $K(v)$ be the corresponding clique.
For $v \notin Z$, let $K(v) = \{v\}$.

We replace every pair (s_i, t_i) with the pairs $K(s_i) \times K(t_i)$.



Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

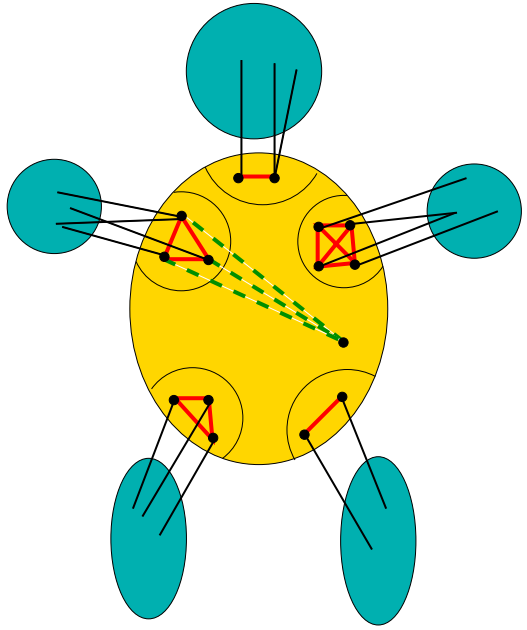
If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

The pairs (s_i, t_i) need to be changed if s_i or t_i is in Z.

For $v \in Z$, let $K(v)$ be the corresponding clique.
For $v \notin Z$, let $K(v) = \{v\}$.

We replace every pair (s_i, t_i) with the pairs $K(s_i) \times K(t_i)$.



Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

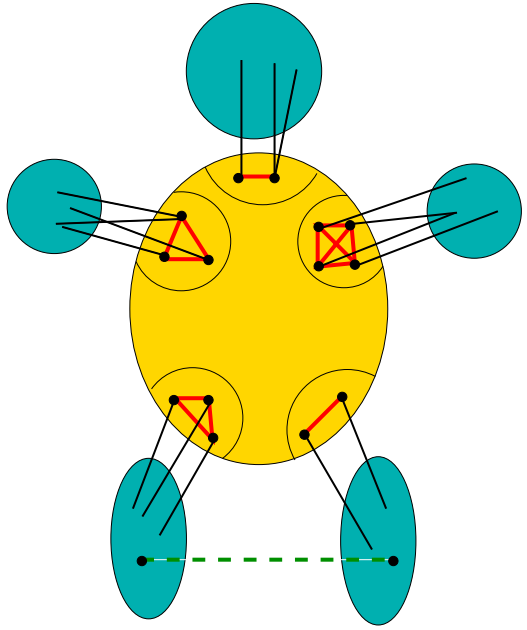
If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

The pairs (s_i, t_i) need to be changed if s_i or t_i is in Z.

For $v \in Z$, let $K(v)$ be the corresponding clique.
For $v \notin Z$, let $K(v) = \{v\}$.

We replace every pair (s_i, t_i) with the pairs $K(s_i) \times K(t_i)$.



Torso of an instance

Let I be a MULTICUT COMPRESSION* instance with graph G.

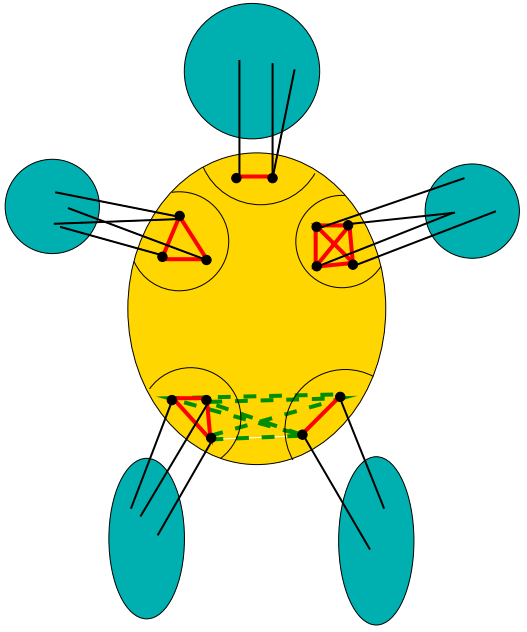
If $Z \subseteq V(G) \setminus W$, then we define a new instance I/Z on the graph torso$(G, V(G) \setminus Z)$.

How do we define the terminal pairs of I/Z?

The pairs (s_i, t_i) need to be changed if s_i or t_i is in Z.

For $v \in Z$, let $K(v)$ be the corresponding clique.
For $v \notin Z$, let $K(v) = \{v\}$.

We replace every pair (s_i, t_i) with the pairs $K(s_i) \times K(t_i)$.



Torso of an instance

Lemma: Let I be an instance of MULTICUT COMPRESSION* and let Z be a set of vertices.

(1) Any solution of I/Z is a solution of I.

(2) If I has a solution S with $S \cap Z = \emptyset$ such that Z covers the isolated part of the solution, then S is a **nonisolating** solution of I/Z.

Torso of an instance

Lemma: Let I be an instance of MULTICUT COMPRESSION* and let Z be a set of vertices.

(1) Any solution of I/Z is a solution of I.

(2) If I has a solution S with $S \cap Z = \emptyset$ such that Z covers the isolated part of the solution, then S is a **nonisolating** solution of I/Z.

So we need to find a Z that is

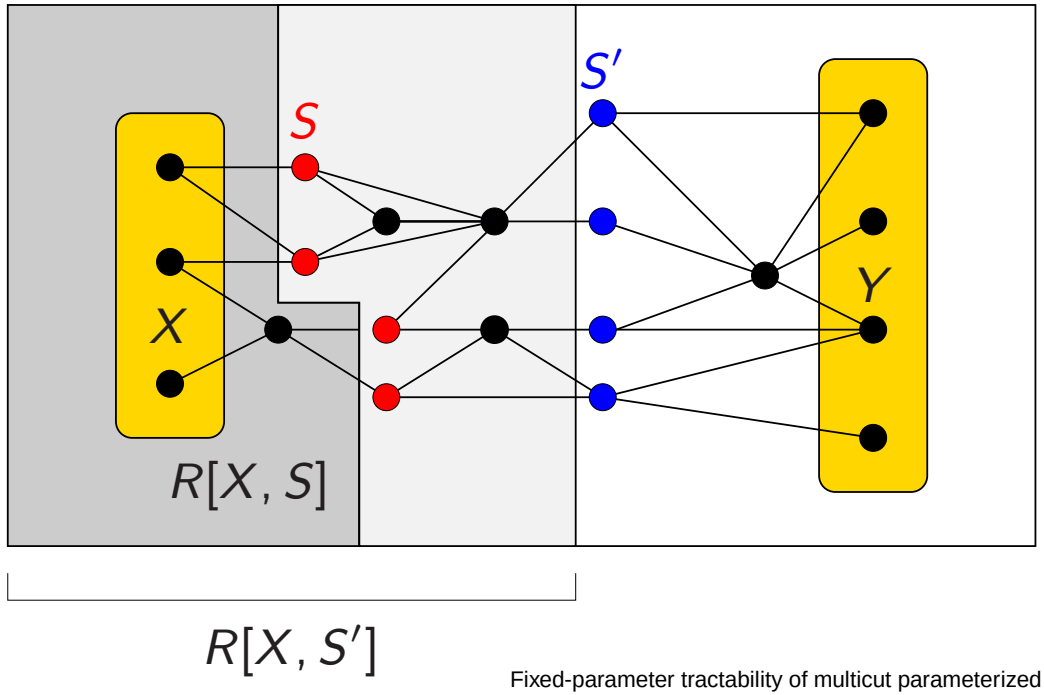
- sufficiently large to cover the isolated part, but
- sufficiently small that it does not contain the (at most p) vertices of S.

Important separators (repeated)

Definition: A set S of vertices is an (X, Y)**-separator** if $S \cap X = S \cap Y = \emptyset$ and there is no $s - t$ path in $G \setminus S$.

Definition: Let $R(X, S)$ be the set of vertices reachable from X in $G \setminus S$.

Definition: An (X, Y)-separator S is **important** if it is inclusionwise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ and $R(X, S) \subset R(X, S')$.



Important components

Definition: A set $C \subseteq V(G) \setminus W$ is an **important component** if $G[C]$ is connected, $|N(C)| \leq p$, and $N(C)$ is an important $C - W$ separator.

In other words: C can be extended only by increasing the size of the neighborhood.

Important components

Definition: A set $C \subseteq V(G) \setminus W$ is an **important component** if $G[C]$ is connected, $|N(C)| \leq p$, and $N(C)$ is an important $C - W$ separator.

In other words: C can be extended only by increasing the size of the neighborhood.

Observation: If $G[C]$ is connected and $|N(C)| \leq p$, then C is an important component iff $N(C)$ is an important $v - W$ separator for every $v \in C$.

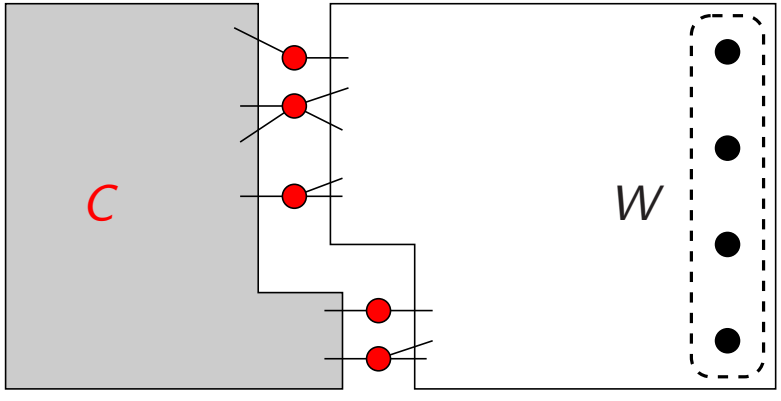
This means that

- Each vertex is contained in at most 4^p important components.
- There are at most $4^p \cdot |V(G)|$ important components and we can enumerate them in time $O^*(4^p)$.

Pushing important components

Lemma: There is a solution S such that every component induced by the isolated part is an important component.

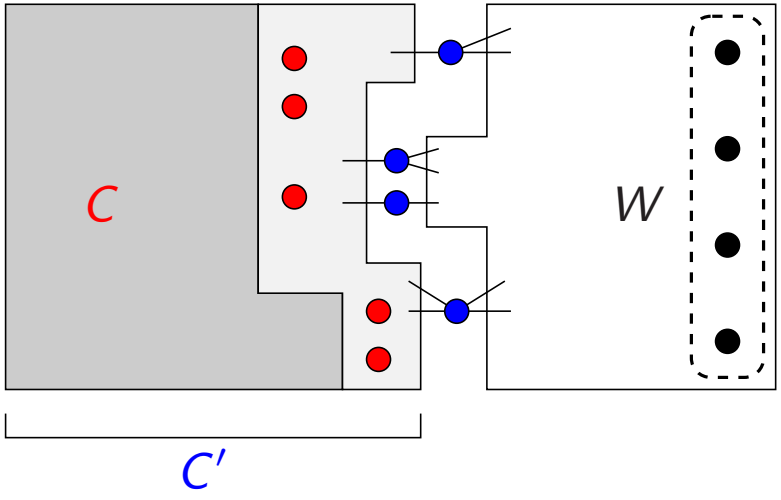
Proof: If C is not an important component, then there is an important component $C' \supset C$ with $|N(C')| \leq |N(C)|$. Let $S^* := (S \setminus N(C)) \cup N(C') \Rightarrow |S^*| \leq |S|$



Pushing important components

Lemma: There is a solution S such that every component induced by the isolated part is an important component.

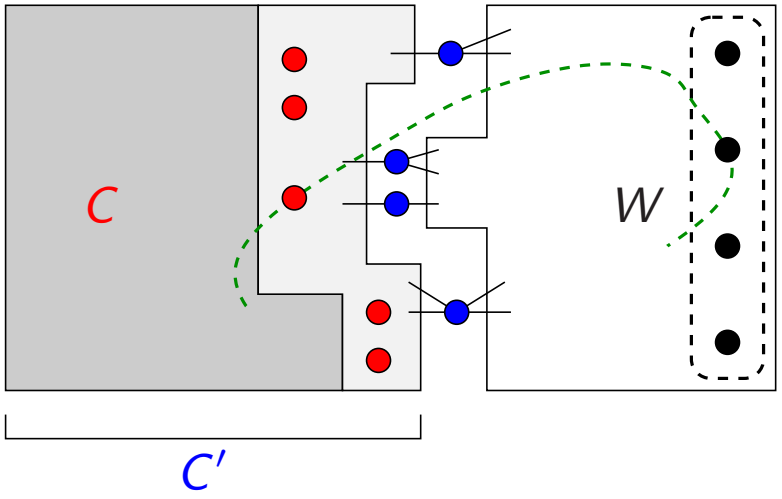
Proof: If C is not an important component, then there is an important component $C' \supset C$ with $|N(C')| \leq |N(C)|$. Let $S^* := (S \setminus N(C)) \cup N(C') \Rightarrow |S^*| \leq |S|$



Pushing important components

Lemma: There is a solution S such that every component induced by the isolated part is an important component.

Proof: If C is not an important component, then there is an important component $C' \supset C$ with $|N(C')| \leq |N(C)|$. Let $S^* := (S \setminus N(C)) \cup N(C') \Rightarrow |S^*| \leq |S|$



S^* remains a solution: problems can be caused only by paths that go through W and a vertex $v \in N(C) \setminus N(C')$. But v is separated from W by $N(C')$.

Important components

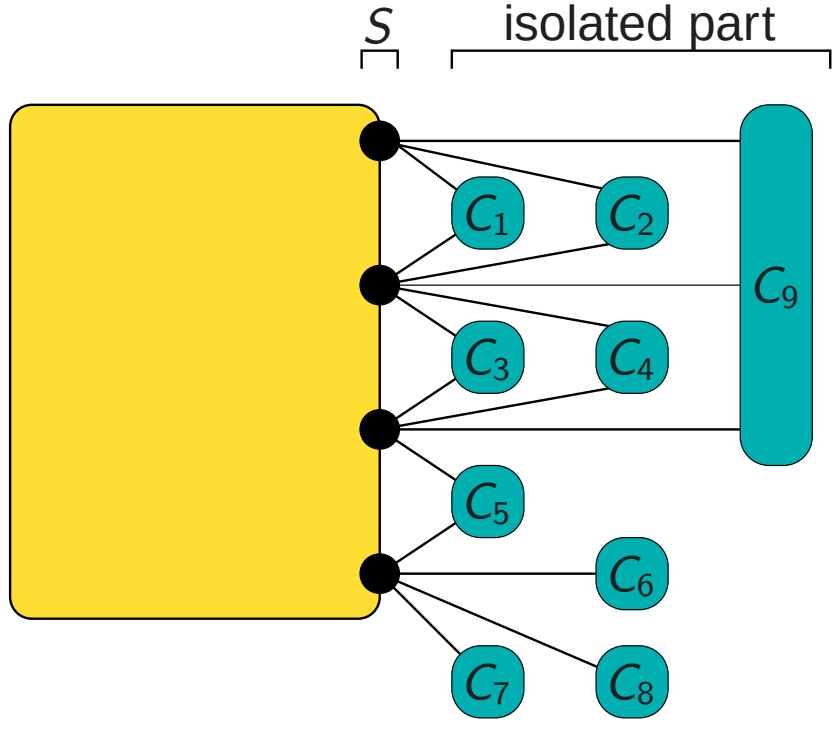
Lemma: There is a solution S such that every component induced by the isolated part is an important component.

This means that we can construct the set Z as the union of important components. However,

- it is not true that the number of important components is at most a constant depending on p (we have only the bound $4^p \cdot |V(G)|$),
- it is not true that the isolated part can be covered by a constant number of important components.

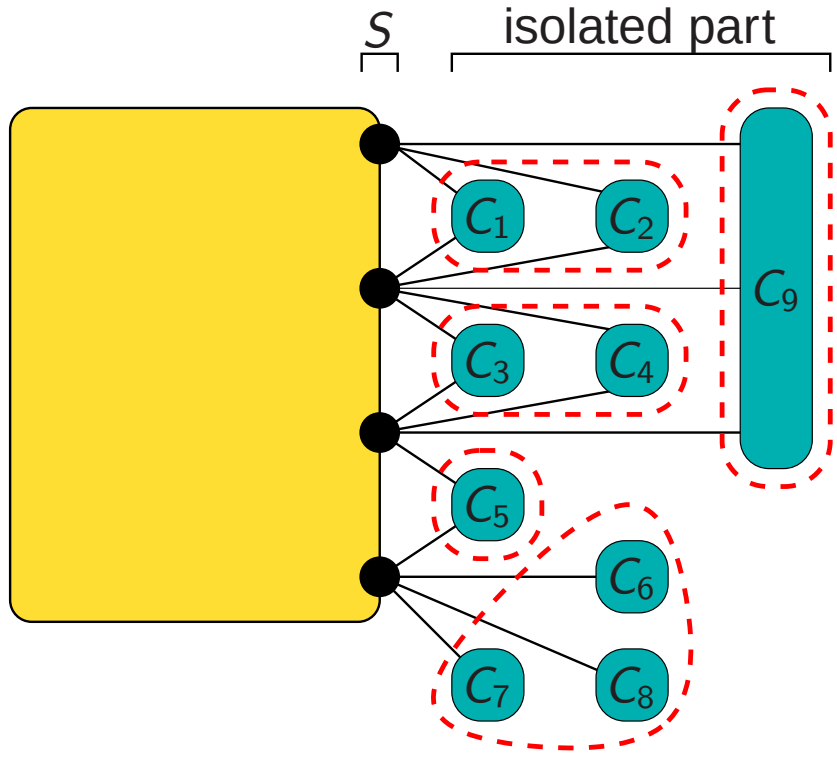
The second problem is minor: we solve it by grouping together components that have the same neighborhood.

Important clusters



Important clusters

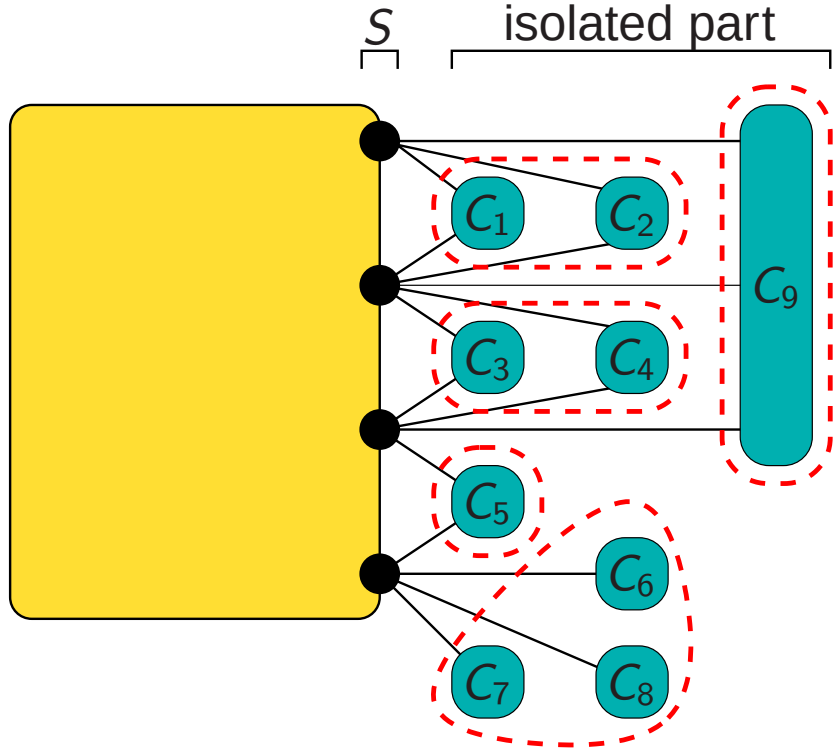
Definition: The **important cluster** L_S is the union of every important component C with $N(C) = S$.



Important clusters

Definition: The **important cluster** L_S is the union of every important component C with $N(C) = S$.

Lemma: There is a solution S such that the isolated part is the union of at most 2^p important clusters.



Random selection of clusters

Lemma: There is a solution S such that the isolated part is the union of at most 2^p important clusters.

The number of important clusters is potentially large, so we cannot do complete enumeration.

Random selection of clusters

Lemma: There is a solution S such that the isolated part is the union of at most 2^p important clusters.

The number of important clusters is potentially large, so we cannot do complete enumeration.

Instead, we select each important cluster independently with probability $\frac{1}{2}$ and let Z be the union of the selected clusters. Estimate the probability that

- (E1) Z covers the isolated part, and
- (E2) $Z \cap S = \emptyset$.

We have seen that if these events hold, then S is a solution of I/Z and the isolated part is empty.

Random selection of clusters

Instead, we select each important cluster independently with probability $\frac{1}{2}$ and let Z be the union of the selected clusters. Estimate the probability that

- (E1) Z covers the isolated part, and
- (E2) $Z \cap S = \emptyset$.

We have seen that if these events hold, then S is a solution of I/Z and the isolated part is empty.

(E1) Holds if the $\leq 2^p$ important clusters of the isolated part are selected.

(E2) Holds if the $\leq p \cdot 4^p$ important clusters intersecting S are not selected.

Probability of (E1)+(E2):

$$\left(\frac{1}{2}\right)^{2^p} \cdot \left(1-\frac{1}{2}\right)^{p\cdot 4^{-p}} = 2^{-2^{O(p)}} \cdot 2^{-2^{O(p)}}$$

Random selection of clusters

Instead, we select each important cluster independently with probability $\frac{1}{2}$ and let Z be the union of the selected clusters. Estimate the probability that

- (E1) Z covers the isolated part, and
- (E2) $Z \cap S = \emptyset$.

We have seen that if these events hold, then S is a solution of I/Z and the isolated part is empty.

(E1) Holds if the $\leq 2^p$ important clusters of the isolated part are selected.

(E2) Holds if the $\leq p \cdot 4^p$ important clusters intersecting S are not selected.

Probability of (E1)+(E2):

$$\left(\frac{1}{2}\right)^{2^p} \cdot \left(1-\frac{1}{2}\right)^{p\cdot 4^{-p}} = 2^{-2^{O(p)}} \cdot 2^{-2^{O(p)}}$$

After $2^{2^{O(p)}}$ trials, we have at least one good Z with high probability.

Derandomization

Previous slide: We randomly select elements from a universe $\mathcal{U}$ such that the good event is if every member of the a-element collection $\mathcal{A}$ is selected ($a \leq 2^p$) and no member of the b-element collection $\mathcal{B}$ is selected ($b \leq p \cdot 4^p$).

Instead of a random subsets, we go through a deterministic family of subsets.

Derandomization

Previous slide: We randomly select elements from a universe $\mathcal{U}$ such that the good event is if every member of the a-element collection $\mathcal{A}$ is selected ($a \leq 2^p$) and no member of the b-element collection $\mathcal{B}$ is selected ($b \leq p \cdot 4^p$).

Instead of a random subsets, we go through a deterministic family of subsets.

An (n, r, r^2)-splitter is a family of functions $[n] \to [r^2]$ such that for every r-element $X \subseteq [n]$, it contains a function that is injective on X.

Theorem: [Naor, Schulman, Srinivasan 1995] There is an explicit construction of an (n, r, r^2)-splitter family containing $O(r^6 \log r \log n)$ functions.

Derandomization

Previous slide: We randomly select elements from a universe $\mathcal{U}$ such that the good event is if every member of the a-element collection $\mathcal{A}$ is selected ($a \leq 2^p$) and no member of the b-element collection $\mathcal{B}$ is selected ($b \leq p \cdot 4^p$).

Instead of a random subsets, we go through a deterministic family of subsets.

An (n, r, r^2)-splitter is a family of functions $[n] \to [r^2]$ such that for every r-element $X \subseteq [n]$, it contains a function that is injective on X.

Theorem: [Naor, Schulman, Srinivasan 1995] There is an explicit construction of an (n, r, r^2)-splitter family containing $O(r^6 \log r \log n)$ functions.

Instead of a random subset of $\mathcal{U}$, we go through every function f of a $(|\mathcal{U}|, a+b, (a+b)^2)$-splitter and every subset F of $[(a+b)^2]$. For a given f, F, we select $x \in \mathcal{U}$ if $f(x) \in F$.

There is an f which is injective on $\mathcal{A} \cup \mathcal{B}$ and an F such that $f(x) \in F$ for every $x \in \mathcal{A}$ and $f(x) \notin F$ for every $x \in \mathcal{B}$.

Improving the probability

We do the random selection in two phases to improve the success probability to $2^{-O(p^3)}$.

Phase 1: Select important clusters with probability 4^{-p} and make the neighborhood of each selected cluster a clique.

$\Rightarrow$ with probability $2^{-O(p^3)}$, S remains a solution and the neighborhood of each component of the isolated part is a clique.

Improving the probability

We do the random selection in two phases to improve the success probability to $2^{-O(p^3)}$.

Phase 1: Select important clusters with probability 4^{-p} and make the neighborhood of each selected cluster a clique.

$\Rightarrow$ with probability $2^{-O(p^3)}$, S remains a solution and the neighborhood of each component of the isolated part is a clique.

Lemma: Each vertex is contained in at most p important clusters whose boundary is a clique.

Phase 2: Select important clusters whose neighborhood is a clique with probability $1 - 2^{-p}$.

With probability $2^{-O(p^3)}$, the $\leq 2^p$ important clusters covering the solution are selected, the $\leq p \cdot p$ important clusters intersecting S are not selected.

Review

We have seen that MULTICUT COMPRESSION* can be solved in time $O^*(15^p)$ by reduction to ALMOST 2SAT if the following two conditions hold:

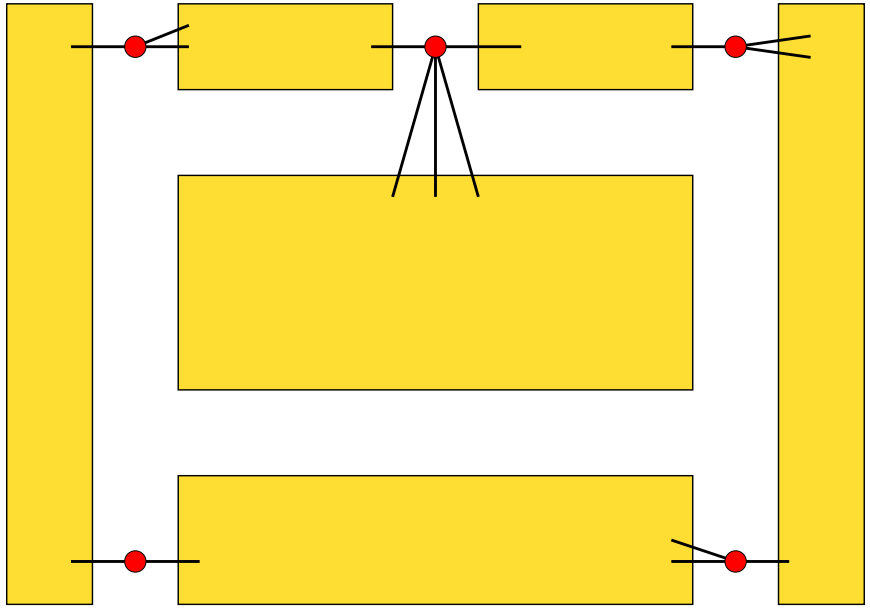
- (1) There is a solution where the isolated part is empty (“nonisolating solution”).
- (2) Every component of $G \setminus W$ has at most two legs, i.e, adjacent to at most two vertices of W (“bipedal instance”).

We have seen how to achieve (1) by random selection of important components.

Next we show how to achieve (2).

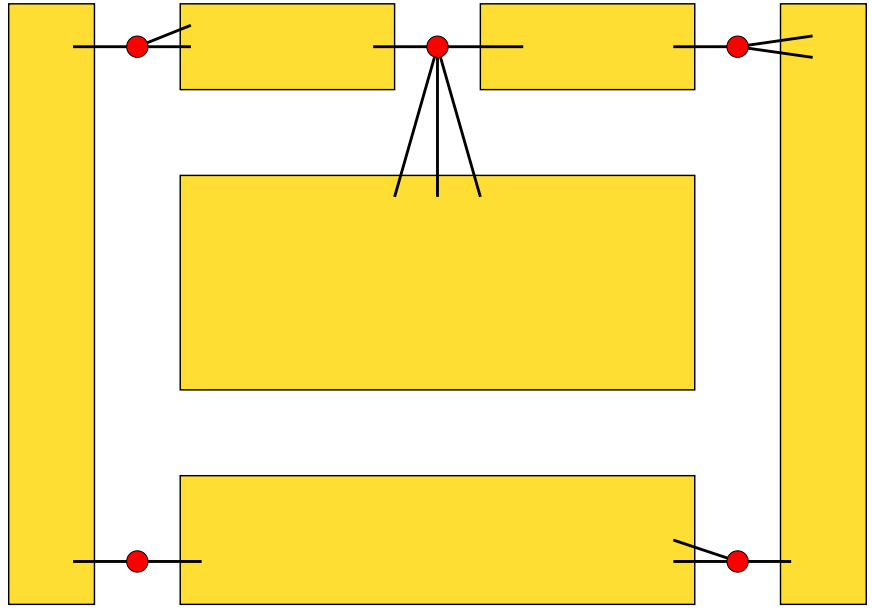
Reduction to the bipedal case

We want to achieve that each component of $G \setminus W$ has at most two legs.



Reduction to the bipedal case

We want to achieve that each component of $G \setminus W$ has at most two legs.



A **nontrival component** is a component having at least two legs. If there are more than p nontrivial components, then there is no solution.

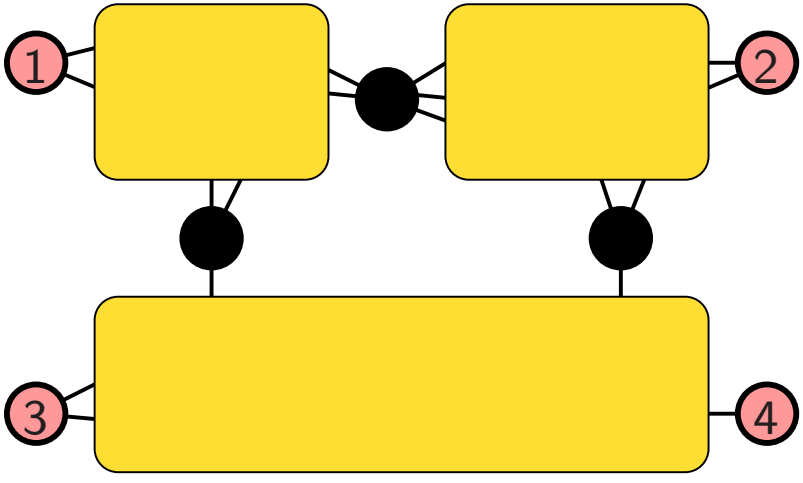
We show that if there is a component having at least 3 legs, then we can increase the number of nontrivial components.

Graceful sets

Consider a component K of $G \setminus W$ having at least 3 legs, and consider some set $B \subseteq K$. We guess what happens to each vertex of B in the solution.

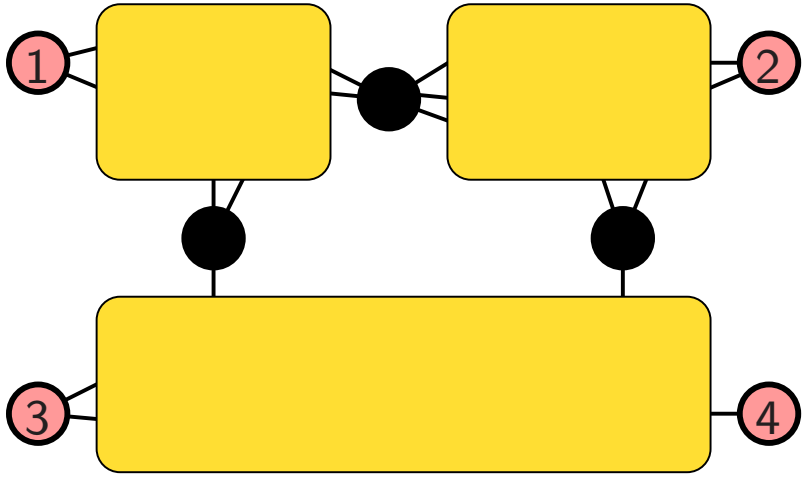
Graceful sets

Consider a component K of $G \setminus W$ having at least 3 legs, and consider some set $B \subseteq K$. We guess what happens to each vertex of B in the solution.



Graceful sets

Consider a component K of $G \setminus W$ having at least 3 legs, and consider some set $B \subseteq K$. We guess what happens to each vertex of B in the solution.



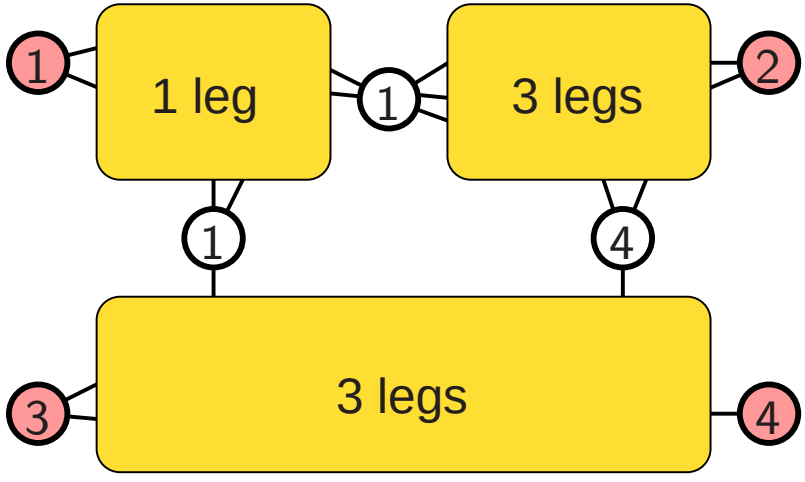
Each vertex is either

- in the solution $\Rightarrow$ delete it and decrease p, or
- reachable from one of the legs $\Rightarrow$ identify the two vertices.

We want to select B such that in every branch where no vertex is deleted, the number of nontrivial components increases.

Graceful sets

Consider a component K of $G \setminus W$ having at least 3 legs, and consider some set $B \subseteq K$. We guess what happens to each vertex of B in the solution.



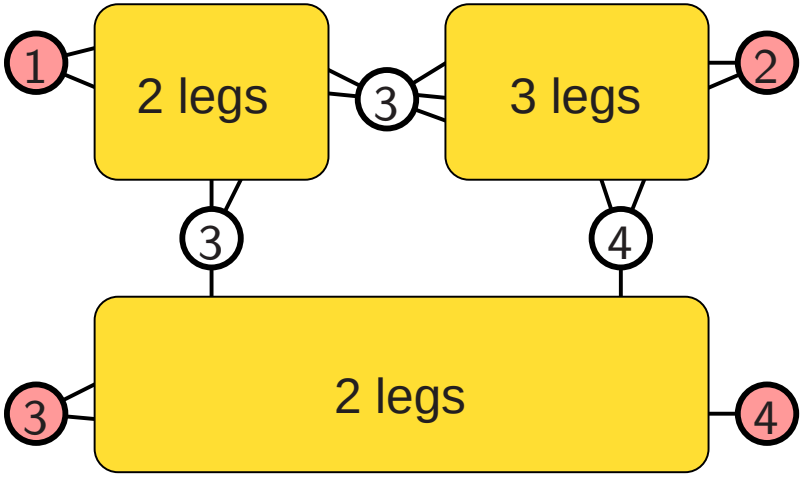
Each vertex is either

- in the solution $\Rightarrow$ delete it and decrease p, or
- reachable from one of the legs $\Rightarrow$ identify the two vertices.

We want to select B such that in every branch where no vertex is deleted, the number of nontrivial components increases.

Graceful sets

Consider a component K of $G \setminus W$ having at least 3 legs, and consider some set $B \subseteq K$. We guess what happens to each vertex of B in the solution.



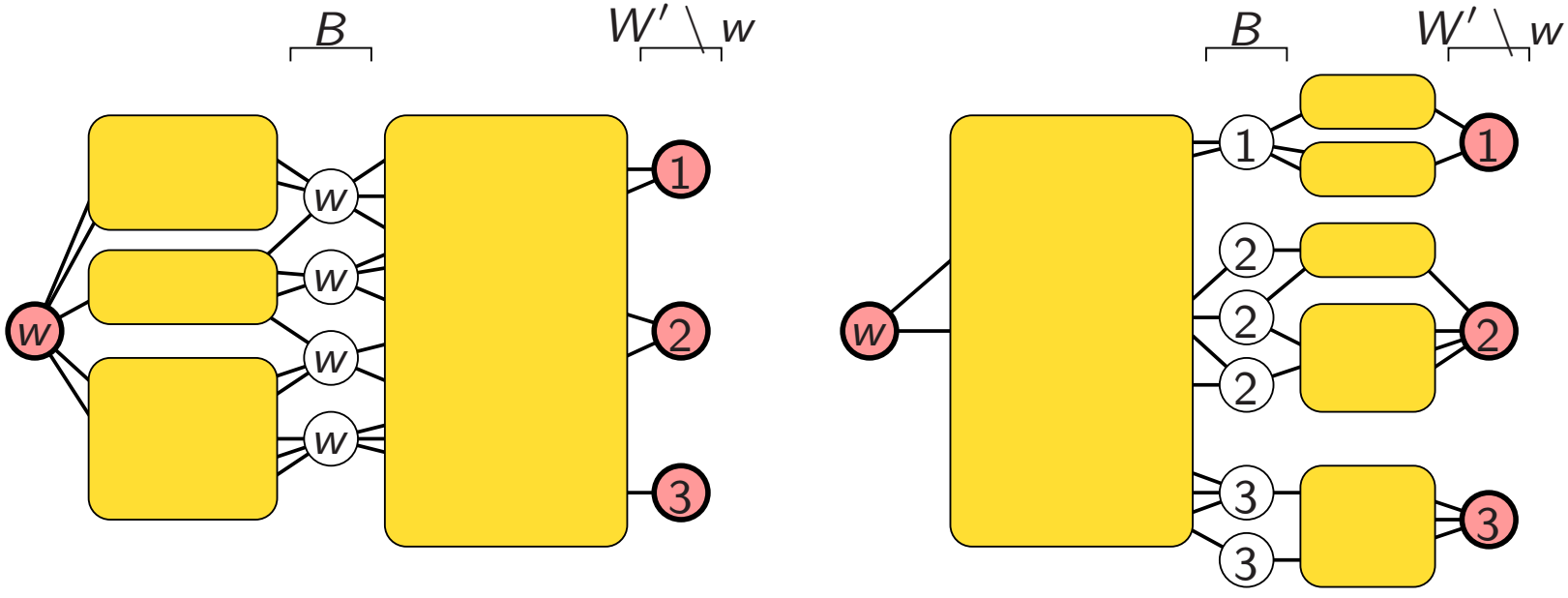
Set B is **graceful** if no matter how we identify the vertices of B with the legs, the number of nontrivial components increases.

Lemma: If there is a component with at least 3 legs, then we can find a graceful set of size $3p$ in polynomial time.

Graceful sets

Let K be a component of $G \setminus W$ with legs W', $|W'| \geq 3$. Let $w \in W'$ and let B be a minimum $w - (W' \setminus w)$ separator.

Then B is a graceful set, except in the following two cases:



In this case, let us continue finding a graceful set inside the “big component.”

Summary of the algorithm

- Creating a nonisolating solution: random selection of important clusters and then taking the torso of the graph.
- Reduction to the bipedal case: selecting graceful sets and then branching on what happens in the set.
- Reduction to ALMOST 2SAT: variables express which leg is reachable from a vertex, deletion of variables and vertices correspond naturally.
- Derandomization is possible.