Perfect Matchings on 3-Regular, Bridgeless Graphs

Marcelo Siqueira

DMAT-UFRN

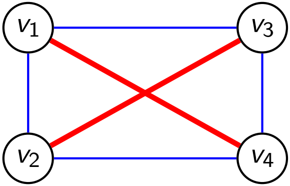
mfsiqueira@mat.ufrn.br

Preliminaries

- Let $G = (V, E)$ be a finite and undirected graph.

Preliminaries

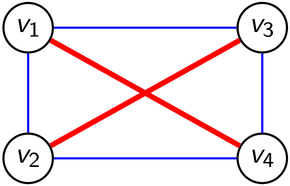
- Let $G = (V, E)$ be a finite and undirected graph.
- A *matching* M on G is any subset of the set E of edges of G such that no two edges of M share a vertex in the set V of vertices of G.



$$M = \{\{v_1, v_4\}, \{v_2, v_3\}\}$$

Preliminaries

- Let $G = (V, E)$ be a finite and undirected graph.
- A *matching* M on G is any subset of the set E of edges of G such that no two edges of M share a vertex in the set V of vertices of G.

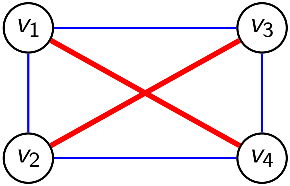


$$M = \{\{v_1, v_4\}, \{v_2, v_3\}\}$$

- Edges in M are called *matching edges*.

Preliminaries

- Let $G = (V, E)$ be a finite and undirected graph.
- A *matching* M on G is any subset of the set E of edges of G such that no two edges of M share a vertex in the set V of vertices of G.



$$M = \{\{v_1, v_4\}, \{v_2, v_3\}\}$$

- Edges in M are called *matching edges*.
- Vertices of matching edges are said to be *matched* or *covered* by M.

Preliminaries

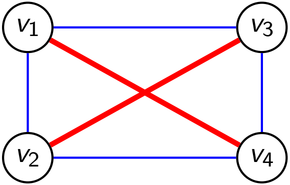
- A matching M on G is a *maximum cardinality* matching on G if and only if $|M| \geq |M'|$, where M' is any possible matching M' on G.

Preliminaries

- A matching M on G is a *maximum cardinality* matching on G if and only if $|M| \geq |M'|$, where M' is any possible matching M' on G. The set

$$M = \{\{v_1, v_4\}, \{v_2, v_3\}\}$$

is a maximum cardinality matching on

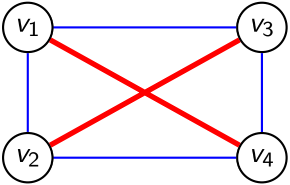


Preliminaries

- A matching M on G is a *maximum cardinality* matching on G if and only if $|M| \geq |M'|$, where M' is any possible matching M' on G. The set

$$M = \{\{v_1, v_4\}, \{v_2, v_3\}\}$$

is a maximum cardinality matching on



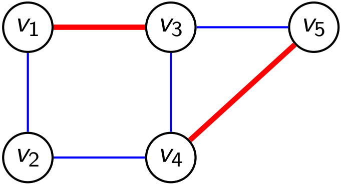
- A matching M on G is said to be *perfect* if and only if all vertices of G are matched by M. So, the matching M in the above example is perfect.

Preliminaries

- Every perfect matching is a maximum cardinality matching.

Preliminaries

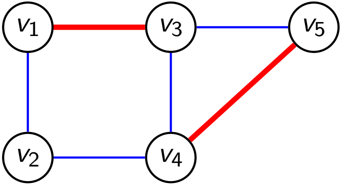
- Every perfect matching is a maximum cardinality matching.
- The reciprocal is not true:



$$M = \{\{v_1, v_3\}, \{v_4, v_5\}\}$$

Preliminaries

- Every perfect matching is a maximum cardinality matching.
- The reciprocal is not true:

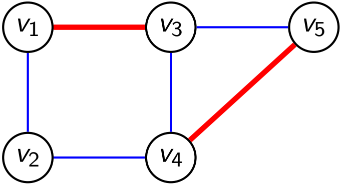


$$M = \{\{v_1, v_3\}, \{v_4, v_5\}\}$$

- There are graphs that always admit perfect matchings (and they show up in graphics applications):

Preliminaries

- Every perfect matching is a maximum cardinality matching.
- The reciprocal is not true:



$$M = \{\{v_1, v_3\}, \{v_4, v_5\}\}$$

- There are graphs that always admit perfect matchings (and they show up in graphics applications): the so-called *3-regular and bridgeless graphs*.

Preliminaries

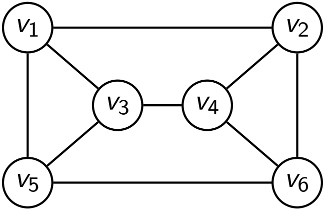
Theorem (Petersen, 1891)

Every 3-regular and bridgeless graph admits a perfect matching.

Preliminaries

Theorem (Petersen, 1891)

Every 3-regular and bridgeless graph admits a perfect matching.

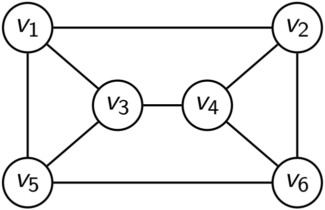


- G is *3-regular* if and only if every vertex of G has degree 3.

Preliminaries

Theorem (Petersen, 1891)

Every 3-regular and bridgeless graph admits a perfect matching.

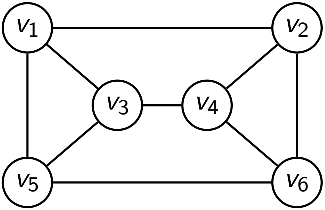


- G is *3-regular* if and only if every vertex of G has degree 3.
- Recall that an edge e of a graph G is said to be a *bridge* (or a *cut edge*) of G if and only if $G - e$ has more connected components than G.

Preliminaries

Theorem (Petersen, 1891)

Every 3-regular and bridgeless graph admits a perfect matching.

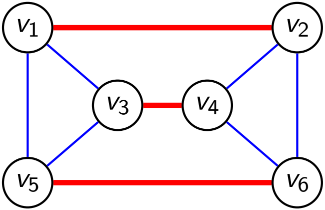


- G is *3-regular* if and only if every vertex of G has degree 3.
- Recall that an edge e of a graph G is said to be a *bridge* (or a *cut edge*) of G if and only if $G - e$ has more connected components than G.
- G is bridgeless if and only if G has no bridges.

Preliminaries

Theorem (Petersen, 1891)

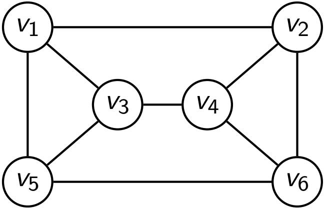
Every 3-regular and bridgeless graph admits a perfect matching.



- G is *3-regular* if and only every vertex of G has degree 3.
- Recall that an edge e of a graph G is said to be a *bridge* (or a *cut edge*) of G if and only if $G - e$ has more connected components than G.
- G is bridgeless if and only if G has no bridges.

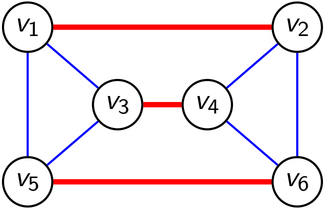
Problem Statement

- **Input**:
  A 3-regular, bridgeless graph $G = (V, E)$.



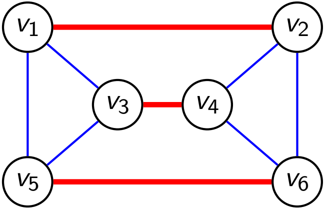
Problem Statement

- **Input**:
  A 3-regular, bridgeless graph $G = (V, E)$.
- **Output**:
  A perfect matching M on G.



Problem Statement

- **Input**:
  A 3-regular, bridgeless graph $G = (V, E)$.
- **Output**:
  A perfect matching M on G.
- **Assumption**:
  G is finite and connected.



A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m, n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m, n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.
    - $\mathcal{O}(mn)$ by Gabow and Tarjan (J. of ACM, 1991).

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m, n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.
    - $\mathcal{O}(mn)$ by Gabow and Tarjan (J. of ACM, 1991).
- New ideas incorporated by Micali and Vazirani yielded the best known upper bound for the blossom-shrinking algorithm: $\mathcal{O}(m\sqrt{n})$ (FOCS, 1980).

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m, n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.
    - $\mathcal{O}(mn)$ by Gabow and Tarjan (J. of ACM, 1991).
- New ideas incorporated by Micali and Vazirani yielded the best known upper bound for the blossom-shrinking algorithm: $\mathcal{O}(m\sqrt{n})$ (FOCS, 1980).
- For cubic graphs, $m = \Theta(n)$. So, we can compute a perfect matching on cubic, bridgeless graphs in $\mathcal{O}(n^{1.5})$ using an algorithm for *general* graphs!

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m,n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.
    - $\mathcal{O}(mn)$ by Gabow and Tarjan (J. of ACM, 1991).
- New ideas incorporated by Micali and Vazirani yielded the best known upper bound for the blossom-shrinking algorithm: $\mathcal{O}(m\sqrt{n})$ (FOCS, 1980).
- For cubic graphs, $m = \Theta(n)$. So, we can compute a perfect matching on cubic, bridgeless graphs in $\mathcal{O}(n^{1.5})$ using an algorithm for *general* graphs!
- Can we do better?

A Bit of History

- J. Edmonds' *blossom-shrinking algorithm* (CJM, 1965).
  - Original algorithm finds maximum cardinality matchings on **general** graphs in $\mathcal{O}(n^4)$ time, where n is the number, $|V|$, of vertices of G.
  - Careful implementations can lower the above upper bound:
    - $\mathcal{O}(n^3)$ by Gabow (J. of ACM, 1976).
    - $\mathcal{O}(mn\alpha(m, n))$ by Tarjan (1985), where m is the number, $|E|$, of edges of G.
    - $\mathcal{O}(mn)$ by Gabow and Tarjan (J. of ACM, 1991).
- New ideas incorporated by Micali and Vazirani yielded the best known upper bound for the blossom-shrinking algorithm: $\mathcal{O}(m\sqrt{n})$ (FOCS, 1980).
- For cubic graphs, $m = \Theta(n)$. So, we can compute a perfect matching on cubic, bridgeless graphs in $\mathcal{O}(n^{1.5})$ using an algorithm for *general* graphs!
- Can we do better? **YES**.

A Bit of History

- The main idea behind the previous algorithms is to try to find an *augmenting path* on G with respect to a *current* matching M on G.

A Bit of History

- The main idea behind the previous algorithms is to try to find an *augmenting path* on G with respect to a *current* matching M on G.

Theorem (Berge, 1957)

Let G be a graph such that G has no loops. Let M be any matching on G. Then, M is a maximum cardinality matching on G if and only if G has no augmenting path with respect to M.

A Bit of History

- The main idea behind the previous algorithms is to try to find an *augmenting path* on G with respect to a *current* matching M on G.

Theorem (Berge, 1957)

Let G be a graph such that G has no loops. Let M be any matching on G. Then, M is a maximum cardinality matching on G if and only if G has no augmenting path with respect to M.

- All improvements on the upper bound of the original Edmonds' blossom-shrinking algorithm are related on how "efficiently" an augmenting path (with respect to the current graph) could be found, if any.

A Bit of History

- The main idea behind the previous algorithms is to try to find an *augmenting path* on G with respect to a *current* matching M on G.

Theorem (Berge, 1957)

Let G be a graph such that G has no loops. Let M be any matching on G. Then, M is a maximum cardinality matching on G if and only if G has no augmenting path with respect to M.

- All improvements on the upper bound of the original Edmonds' blossom-shrinking algorithm are related on how "efficiently" an augmenting path (with respect to the current graph) could be found, if any.
- Note: the *best known upper bound* has been out there for about 35 years!

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.
- The new "paradigm" comes from a constructive proof given by Frink Jr. in 1926 for Petersen's theorem (original proof has some mistakes).

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.
- The new "paradigm" comes from a constructive proof given by Frink Jr. in 1926 for Petersen's theorem (original proof has some mistakes).
- From now on, assume that G is a connected, 3-regular, bridgeless graph.

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.
- The new "paradigm" comes from a constructive proof given by Frink Jr. in 1926 for Petersen's theorem (original proof has some mistakes).
- From now on, assume that G is a connected, 3-regular, bridgeless graph.
- G is not necessarily *simple*.

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.
- The new "paradigm" comes from a constructive proof given by Frink Jr. in 1926 for Petersen's theorem (original proof has some mistakes).
- From now on, assume that G is a connected, 3-regular, bridgeless graph.
- G is not necessarily *simple*.
- Let e be a *simple* edge (i.e., neither a loop nor a parallel edge) of G.

Frink's Theorem

- It took a completely different *paradigm* to produce an algorithm for the class of 3-regular, cubic graphs with a better upper bound than $\mathcal{O}(n^{1.5})$.
- The new "paradigm" comes from a constructive proof given by Frink Jr. in 1926 for Petersen's theorem (original proof has some mistakes).
- From now on, assume that G is a connected, 3-regular, bridgeless graph.
- G is not necessarily *simple*.
- Let e be a *simple* edge (i.e., neither a loop nor a parallel edge) of G.

Teaser

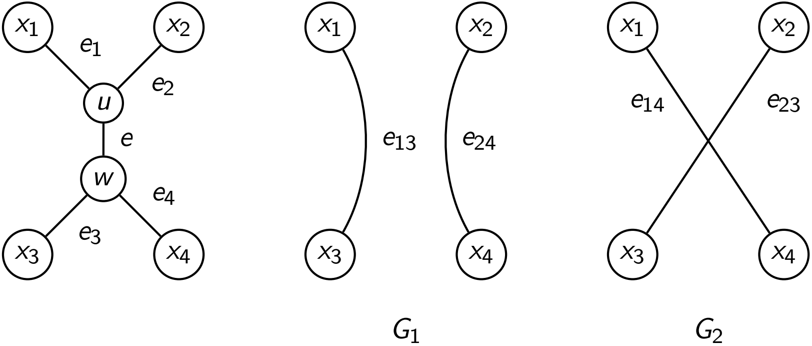
What can you say about G if no edge of G is simple?

Frink's Theorem

- A **reduction operation** is the key idea behind Frink's proof:

Frink's Theorem

- A **reduction operation** is the key idea behind Frink's proof:



Frink's Theorem

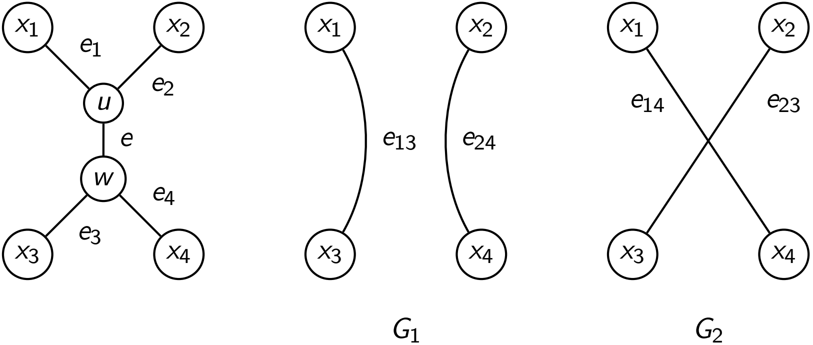
- Frink proved the following:

Frink's Theorem

- Frink proved the following:

Frink's Theorem (Frink, 1926)

At least one of G_1 or G_2 is connected, 3-regular, and bridgeless.



Frink's Algorithm

- So what?

Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.

Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.

Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

Key idea:

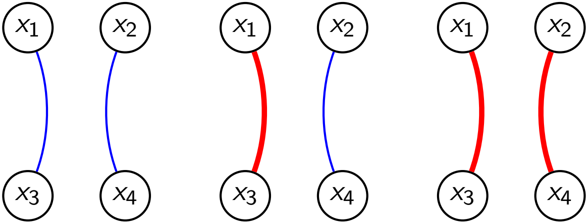
Build a perfect matching M on G from M_1.

Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

Key idea:

Build a perfect matching M on G from M_1.

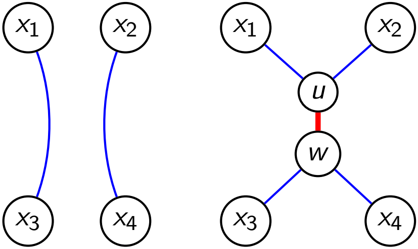


Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

Key idea:

Build a perfect matching M on G from M_1.

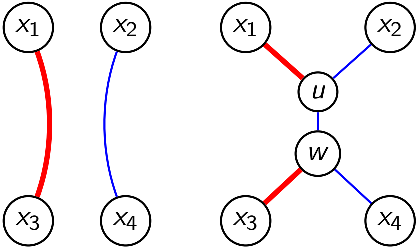


Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

Key idea:

Build a perfect matching M on G from M_1.

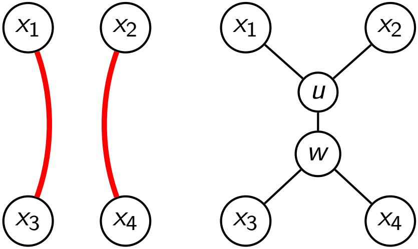


Frink's Algorithm

- So what?
- Suppose G_1 satisfies Frink's theorem.
- By Petersen's theorem, G_1 admits a perfect matching.
- Let M_1 be such a matching.

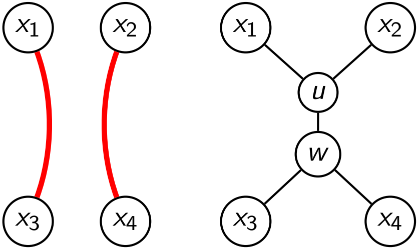
Key idea:

Build a perfect matching M on G from M_1.



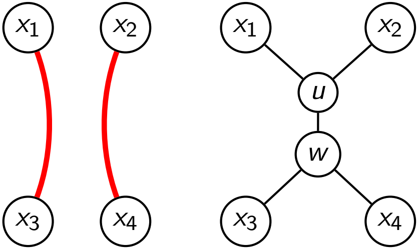
Frink's Algorithm

- How can we solve the third case?



Frink's Algorithm

- How can we solve the third case?

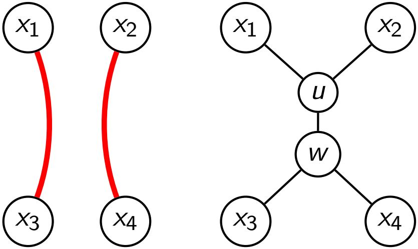


A nice property of matching edges:

Every matching edge belongs to an alternating cycle.

Frink's Algorithm

- How can we solve the third case?



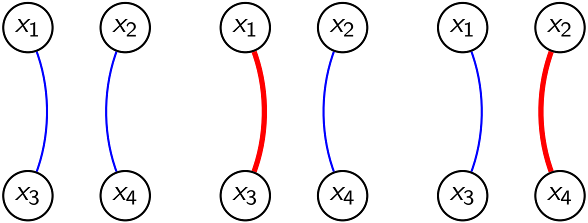
A nice property of matching edges:

Every matching edge belongs to an alternating cycle.

- So, reverse an alternating cycle with either $\{x_1, x_3\}$ or $\{x_2, x_4\}$.

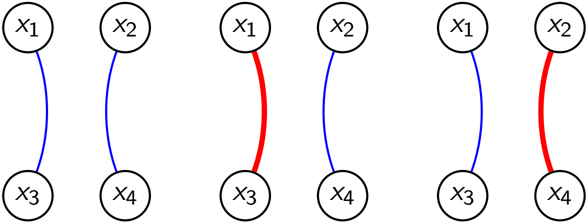
Frink's Algorithm

- Reversing the alternating cycle produces one the following:



Frink's Algorithm

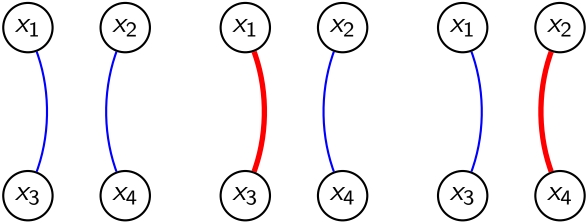
- Reversing the alternating cycle produces one the following:



- Good, but...

Frink's Algorithm

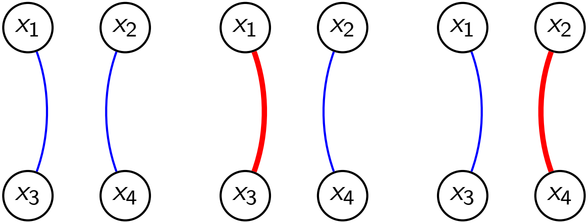
- Reversing the alternating cycle produces one the following:



- Good, but...
  - How can we find the alternating cycle at first place?

Frink's Algorithm

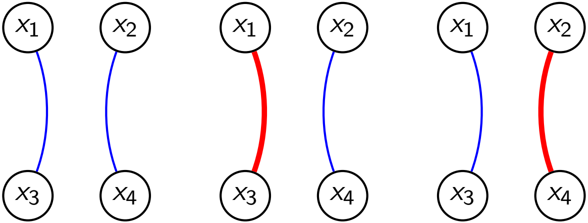
- Reversing the alternating cycle produces one the following:



- Good, but...
  - How can we find the alternating cycle at first place?
  - Finding an augmenting path on $G_1 - \{x_1, x_3\}$ or $G_1 - \{x_2, x_4\}$.

Frink's Algorithm

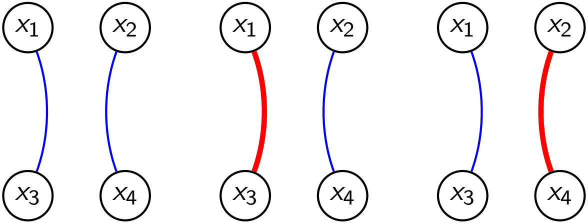
- Reversing the alternating cycle produces one the following:



- Good, but...
  - How can we find the alternating cycle at first place?
  - Finding an augmenting path on $G_1 - \{x_1, x_3\}$ or $G_1 - \{x_2, x_4\}$.
  - What is the cost?

Frink's Algorithm

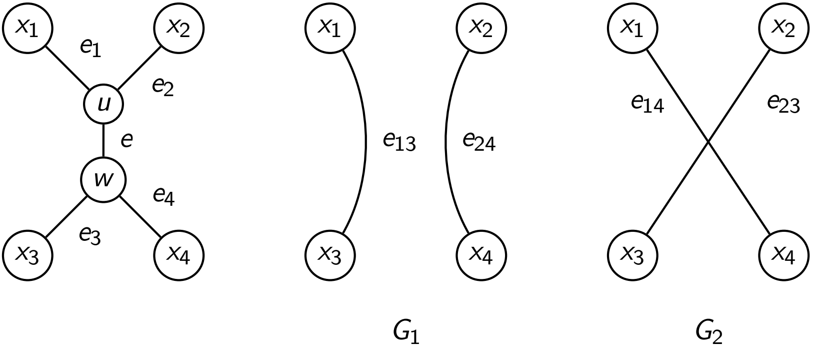
- Reversing the alternating cycle produces one the following:



- Good, but...
  - How can we find the alternating cycle at first place?
  - Finding an augmenting path on $G_1 - \{x_1, x_3\}$ or $G_1 - \{x_2, x_4\}$.
  - What is the cost?
  - $\mathcal{O}(m)$ amortized time (**be careful here**!)

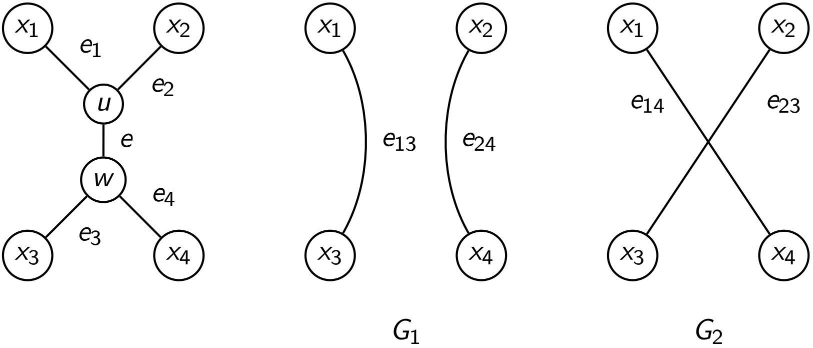
Frink's Algorithm

- A "little" problem remains unsolved:



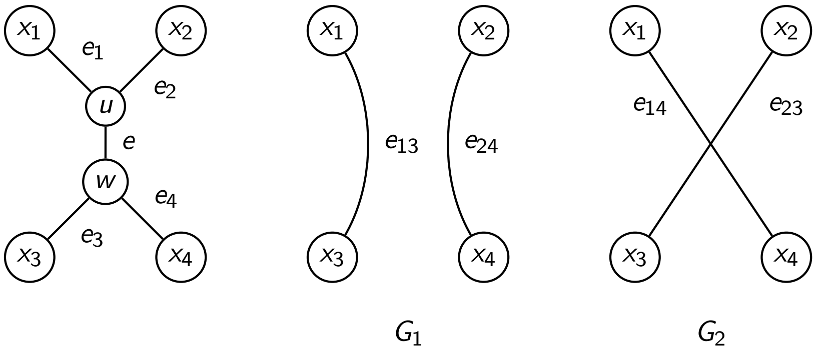
Frink's Algorithm

- A "little" problem remains unsolved:
- How can we decide which graph (G_1 or G_2) satisfies Frink's theorem?



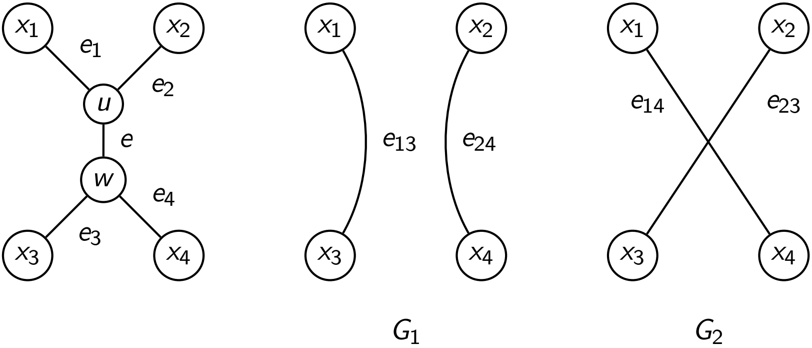
Frink's Algorithm

- Counting biconnected components of G_1 and G_2.



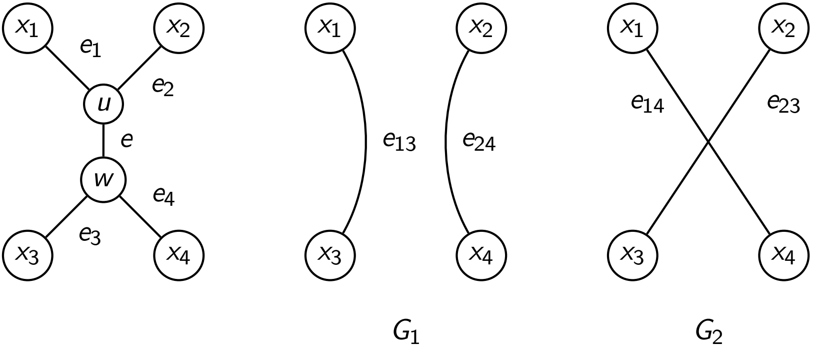
Frink's Algorithm

- Counting biconnected components of G_1 and G_2.
- Can be done with a DFS in $\mathcal{O}(n+m) = \mathcal{O}(n)$ (not the best bound).



Frink's Algorithm

- Counting biconnected components of G_1 and G_2.
- Can be done with a DFS in $\mathcal{O}(n+m) = \mathcal{O}(n)$ (not the best bound).
- So, we can compute a perfect matching on G in $\mathcal{O}(n^2)$ time.



Avoiding Alternating Cycle Reversal

- We can lower the $\mathcal{O}(n^2)$ upper bound to $\mathcal{O}(n \lg^4 n)$ by making two changes in the previous algorithm (Biedl, Bose, Demaine, Lubiw, 2001).

Avoiding Alternating Cycle Reversal

- We can lower the $\mathcal{O}(n^2)$ upper bound to $\mathcal{O}(n \lg^4 n)$ by making two changes in the previous algorithm (Biedl, Bose, Demaine, Lubiw, 2001).
- First, fix an edge f of the *input* graph G such that
  - f is adjacent to the reduction edge before a reduction,

Avoiding Alternating Cycle Reversal

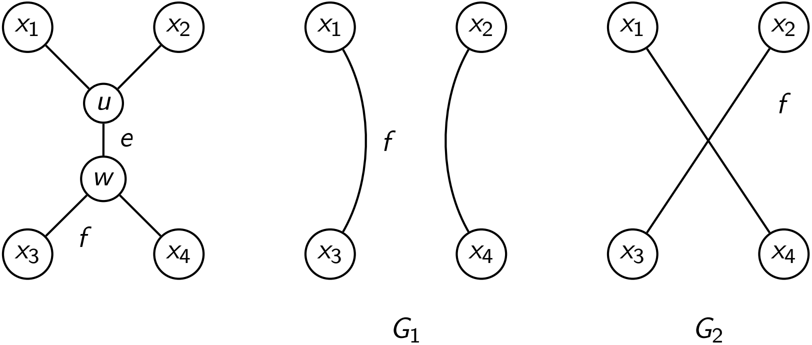
- We can lower the $\mathcal{O}(n^2)$ upper bound to $\mathcal{O}(n \lg^4 n)$ by making two changes in the previous algorithm (Biedl, Bose, Demaine, Lubiw, 2001).
- First, fix an edge f of the *input* graph G such that
  - f is adjacent to the reduction edge before a reduction,
  - f becomes one of the new edges right after the reduction, and

Avoiding Alternating Cycle Reversal

- We can lower the $\mathcal{O}(n^2)$ upper bound to $\mathcal{O}(n \lg^4 n)$ by making two changes in the previous algorithm (Biedl, Bose, Demaine, Lubiw, 2001).
- First, fix an edge f of the *input* graph G such that
  - f is adjacent to the reduction edge before a reduction,
  - f becomes one of the new edges right after the reduction, and
  - f never becomes a matching edge.

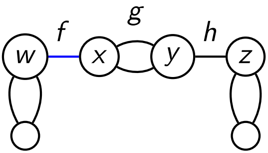
Avoiding Alternating Cycle Reversal

- We can lower the $\mathcal{O}(n^2)$ upper bound to $\mathcal{O}(n \lg^4 n)$ by making two changes in the previous algorithm (Biedl, Bose, Demaine, Lubiw, 2001).
- First, fix an edge f of the *input* graph G such that
  - f is adjacent to the reduction edge before a reduction,
  - f becomes one of the new edges right after the reduction, and
  - f never becomes a matching edge.



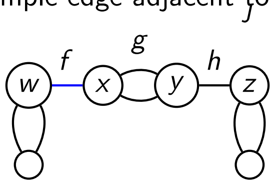
Avoiding Alternating Cycle Reversal

- What if there is no simple edge adjacent to f?

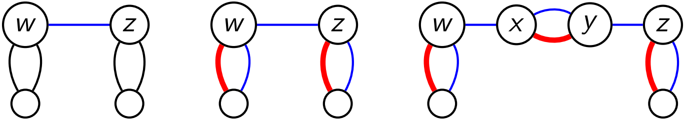


Avoiding Alternating Cycle Reversal

- What if there is no simple edge adjacent to f?

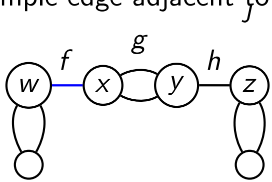


- No big deal...

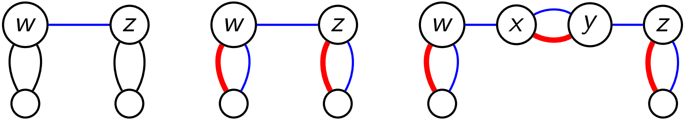


Avoiding Alternating Cycle Reversal

- What if there is no simple edge adjacent to f?



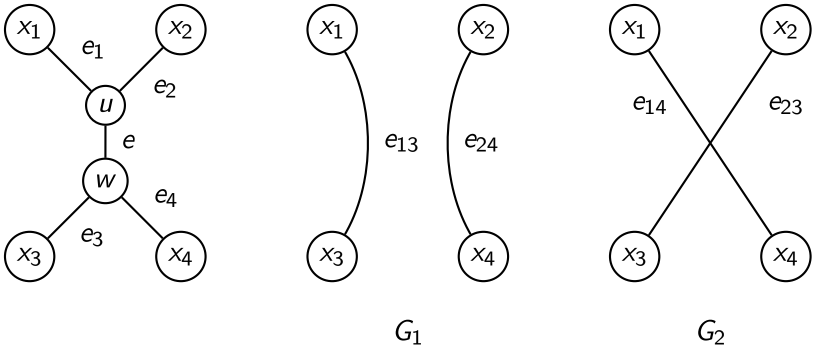
- No big deal...



- So, each reduction takes constant time now!

A Faster Biconnectivity Test

- Recalling...
- How can we decide which graph (G_1 or G_2) satisfies Frink's theorem?



A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better?

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better? **YES**: $\mathcal{O}(n \lg^2 n)$.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better? **YES**: $\mathcal{O}(n \lg^2 n)$.
- Due to Diks and Stanczyk's improvements.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better? **YES**: $\mathcal{O}(n \lg^2 n)$.
- Due to Diks and Stanczyk's improvements.
- A student of mine implemented Diks and Stanczyk's algorithm.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better? **YES**: $\mathcal{O}(n \lg^2 n)$.
- Due to Diks and Stanczyk's improvements.
- A student of mine implemented Diks and Stanczyk's algorithm.
- His source code is freely and publicly available.

A Faster Biconnectivity Test

- Resort to a dynamic connectivity graph data structure (Holm et al., 2001).
- Consider G_1.
- Solve the 2-edge-connectivity problem for each pair in $\{x_1, x_2, x_3, x_4\}$.
- Each test takes $\mathcal{O}(\lg^4 n)$ amortized time.
- This gives us the $\mathcal{O}(n \lg^4 n)$ amortized time upper bound.
- Can we do better? **YES**: $\mathcal{O}(n \lg^2 n)$.
- Due to Diks and Stanczyk's improvements.
- A student of mine implemented Diks and Stanczyk's algorithm.
- His source code is freely and publicly available.
- Can talk about Diks and Stanczyk's algorithm some other time...