An algorithm for constructing a matroid's k-tree

Nick Brettell
Joint work with Charles Semple

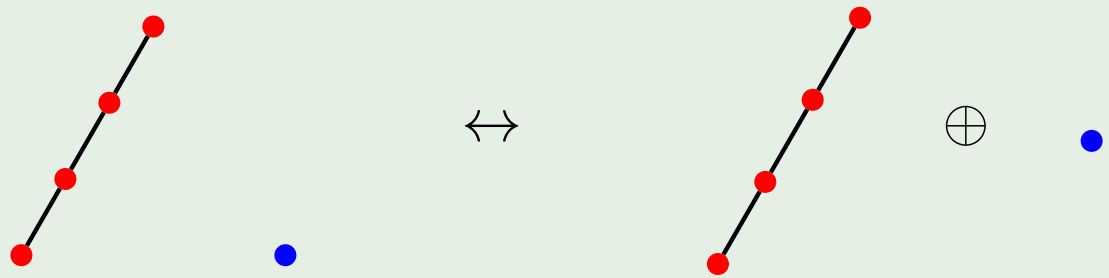
Department of Mathematics and Statistics
University of Canterbury, New Zealand

2014 International Workshop on Structure in Graphs and Matroids
Princeton, July 2014

Matroid decompositions

- We can decompose a matroid into its (2-connected) components (Whitney, 1935)

Example



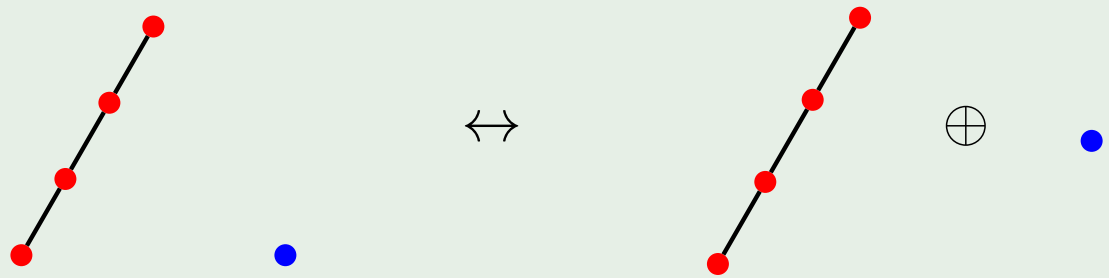
- We can decompose a 2-connected matroid into 3-connected pieces (Cunningham and Edmonds, 1980)

Example

Matroid decompositions

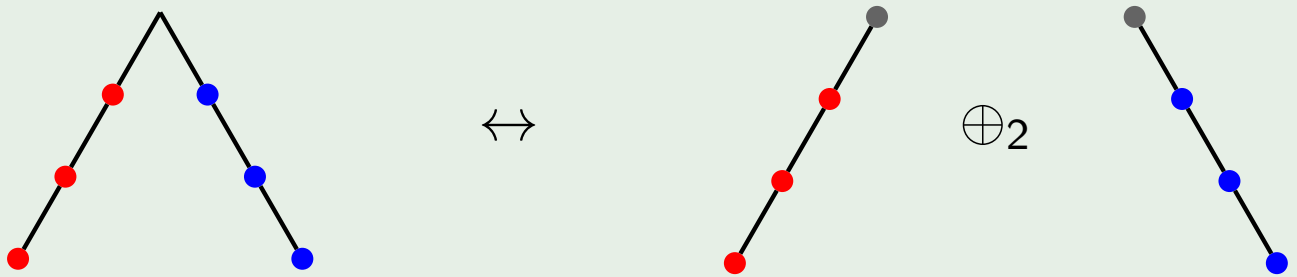
- We can decompose a matroid into its (2-connected) components (Whitney, 1935)

Example



- We can decompose a 2-connected matroid into 3-connected pieces (Cunningham and Edmonds, 1980)

Example



Matroid decompositions (2)

Theorem (Cunningham and Edmonds, 1980)

A 2-connected matroid has a tree decomposition in which every vertex label corresponds to a 3-connected matroid.

These results allow us to apply induction arguments

- Often have: "A matroid has property x if the 3-connected components of the 2-sum decomposition have property x"
- Contributed to initial research focus on 3-connected matroids

Can we decompose a 3-connected matroid into more highly-connected pieces?

Matroid decompositions (2)

Theorem (Cunningham and Edmonds, 1980)

A 2-connected matroid has a tree decomposition in which every vertex label corresponds to a 3-connected matroid.

These results allow us to apply induction arguments

- Often have: "A matroid has property x if the 3-connected components of the 2-sum decomposition have property x"
- Contributed to initial research focus on 3-connected matroids

Can we decompose a 3-connected matroid into more highly-connected pieces?

Matroid decompositions (2)

Theorem (Cunningham and Edmonds, 1980)

A 2-connected matroid has a tree decomposition in which every vertex label corresponds to a 3-connected matroid.

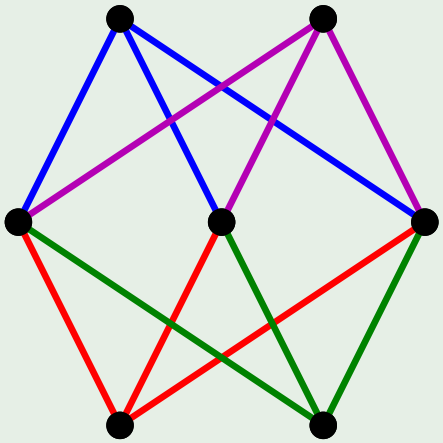
These results allow us to apply induction arguments

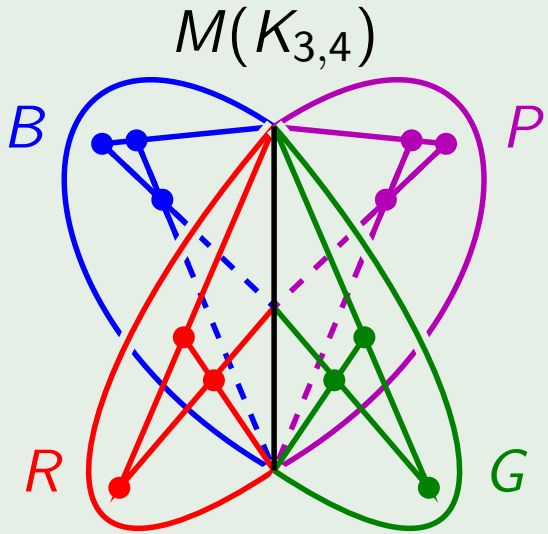
- Often have: "A matroid has property x if the 3-connected components of the 2-sum decomposition have property x"
- Contributed to initial research focus on 3-connected matroids

Can we decompose a 3-connected matroid into more highly-connected pieces?

Crossing 3-separations

Example

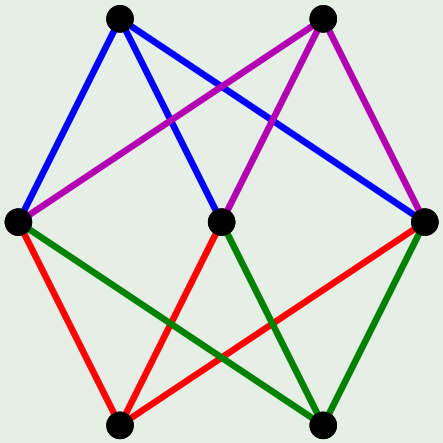


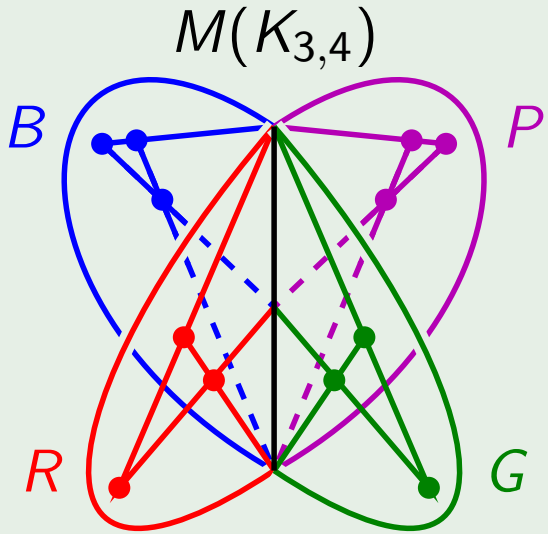


- 3-connected
- Crossing 3-separations: $(R \cup G, B \cup P)$, $(R \cup P, B \cup G)$ and, $(R \cup B, G \cup P)$.

Crossing 3-separations

Example





- 3-connected
- Crossing 3-separations: $(R \cup G, B \cup P)$, $(R \cup P, B \cup G)$ and, $(R \cup B, G \cup P)$.

Equivalent 3-separations

Theorem (Oxley, Semple, and Whittle, 2004)

For a 3-connected matroid M with $|E(M)| \geq 9$, there is a tree that displays, up to a natural equivalence, all the non-sequential 3-separations.

- Called a 3-tree

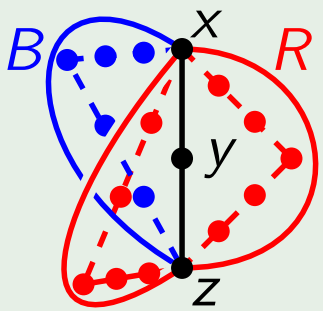
- X is fully closed if it is closed and coclosed
- $\mathrm{fcl}(X)$ is the intersection of all fully closed sets containing X
- (X, Y) is equivalent to (A, B) if $\{\mathrm{fcl}(X), \mathrm{fcl}(Y)\} = \{\mathrm{fcl}(A), \mathrm{fcl}(B)\}$

Equivalent 3-separations

Theorem (Oxley, Semple, and Whittle, 2004)

For a 3-connected matroid M with $|E(M)| \geq 9$, there is a tree that displays, up to a natural equivalence, all the non-sequential 3-separations.

Example



- $(R \cup \{x, y, z\}, B)$ and $(R, B \cup \{x, y, z\})$ are equivalent 3-separations

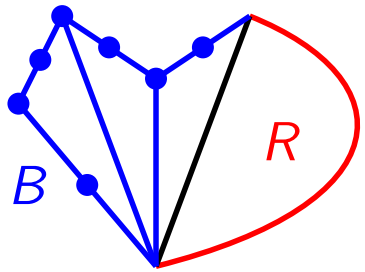
- X is fully closed if it is closed and coclosed
- $\mathrm{fcl}(X)$ is the intersection of all fully closed sets containing X
- (X, Y) is equivalent to (A, B) if $\{\mathrm{fcl}(X), \mathrm{fcl}(Y)\} = \{\mathrm{fcl}(A), \mathrm{fcl}(B)\}$

Equivalent 3-separations (2)

Theorem (Oxley, Semple, and Whittle, 2004)

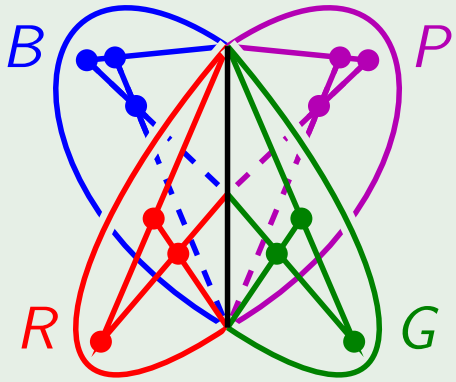
For a 3-connected matroid M, there is a tree that displays, up to a natural equivalence, all the non-sequential 3-separations of M.

- (X, Y) is sequential if $\mathrm{fcl}(X) = E(M)$ or $\mathrm{fcl}(Y) = E(M)$
  - A sequential 3-separation (R, B):

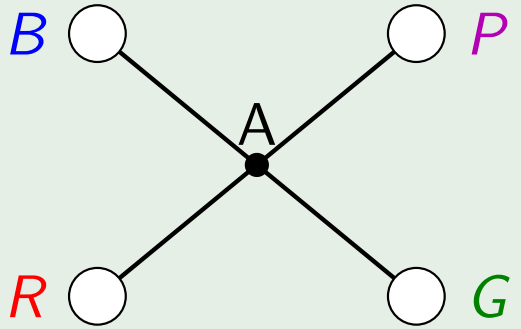


3-trees

Example



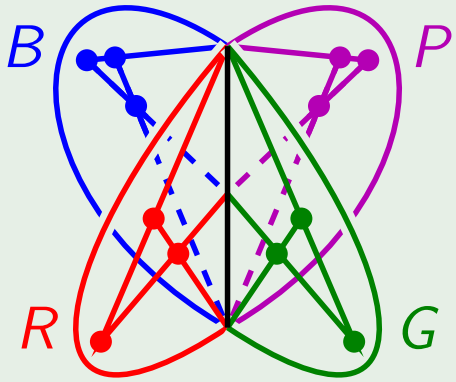
- The associated 3-tree is:



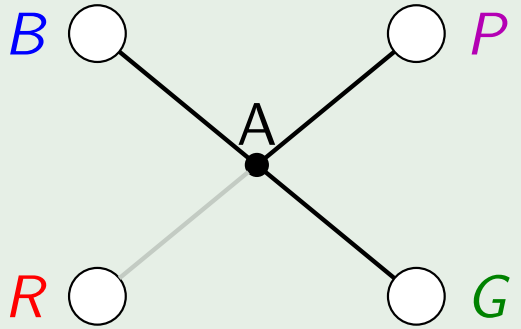
- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$
- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

3-trees

Example



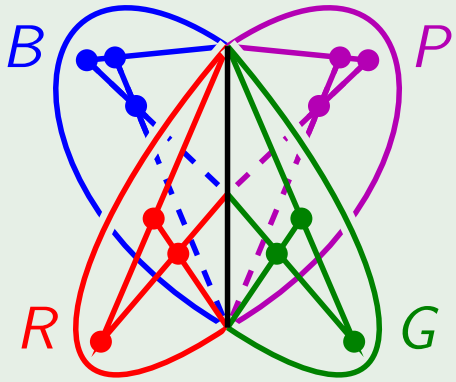
- The associated 3-tree is:



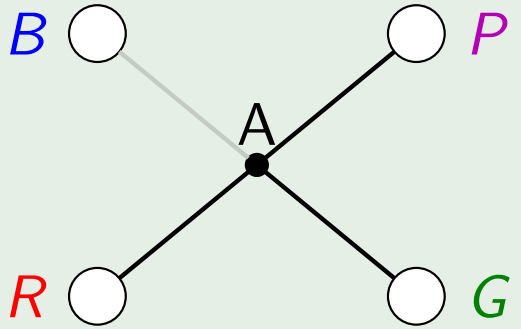
- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$
- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

3-trees

Example



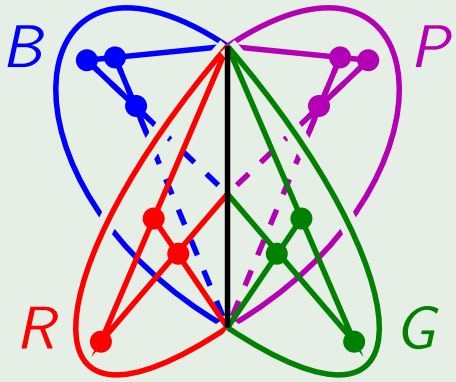
- The associated 3-tree is:



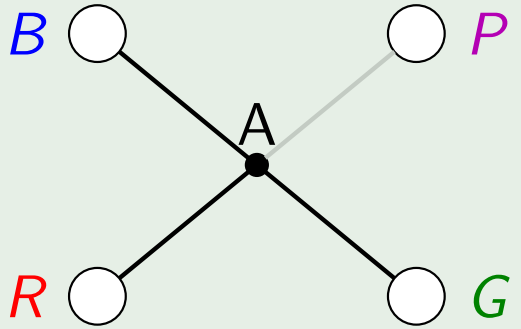
- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$
- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

3-trees

Example



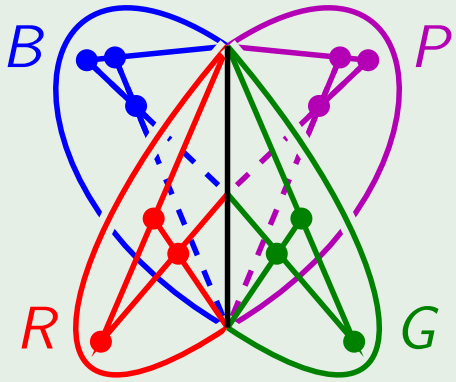
- The associated 3-tree is:



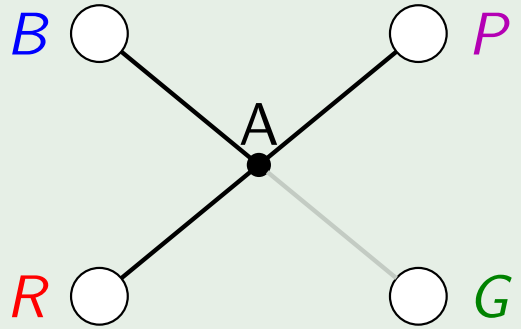
- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$
- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

3-trees

Example



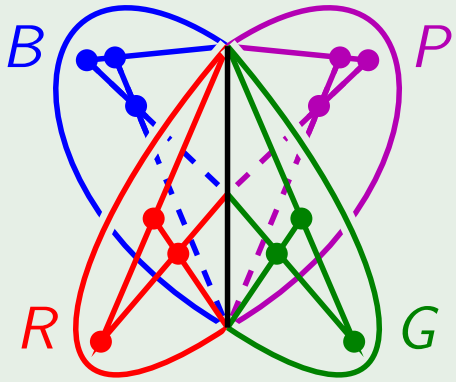
- The associated 3-tree is:



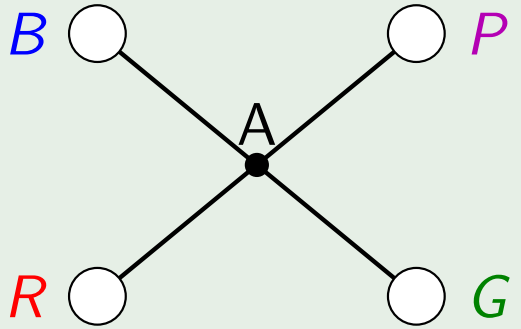
- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$
- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

3-trees

Example



- The associated 3-tree is:



- Each edge corresponds to a 3-separation: $(R, B \cup P \cup G)$, $(B, P \cup G \cup R)$, $(P, G \cup R \cup B)$, and $(G, R \cup B \cup P)$

- (R, B, P, G) is a flower, and R, B, P, and G are the petals
  - A is a flower vertex
  - the others are bag vertices

Flowers

- Flowers fall into two categories: anemones and daisies
- Any partition of petals in an anemone gives a 3-separation

- For a daisy, the petals in each partition must be consecutive

Flowers

- Flowers fall into two categories: anemones and daisies
- Any partition of petals in an anemone gives a 3-separation

- For a daisy, the petals in each partition must be consecutive

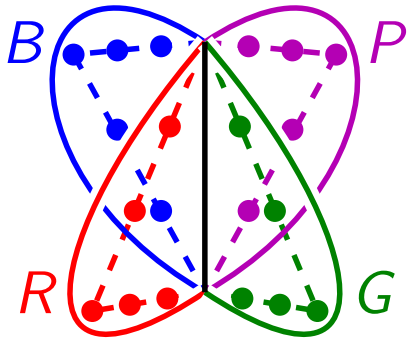
Flowers

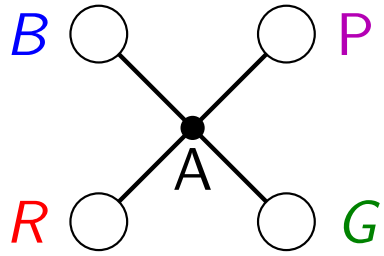
- Flowers fall into two categories: anemones and daisies
- Any partition of petals in an anemone gives a 3-separation

- For a daisy, the petals in each partition must be consecutive

Flowers

- Flowers fall into two categories: anemones and daisies
- Any partition of petals in an anemone gives a 3-separation



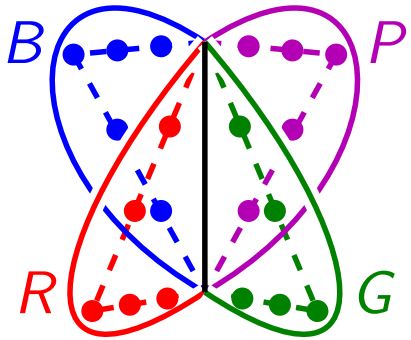


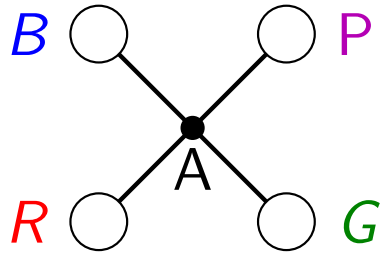
$(B \cup G, P \cup R)$ is a 3-separation

- For a daisy, the petals in each partition must be consecutive

Flowers

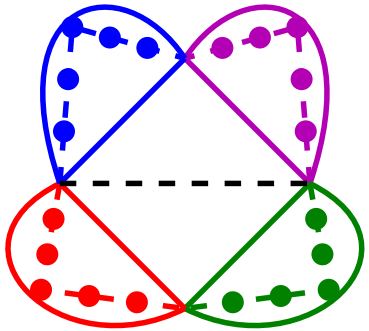
- Flowers fall into two categories: anemones and daisies
- Any partition of petals in an anemone gives a 3-separation

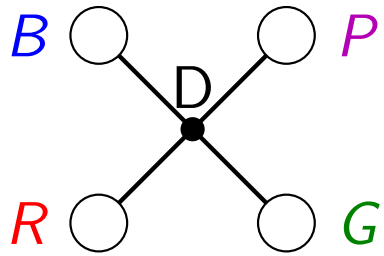




$(B \cup G, P \cup R)$ is a 3-separation

- For a daisy, the petals in each partition must be consecutive





$(B \cup G, P \cup R)$ is not a 3-separation

Decompositions revisited

- A 3-connected representable matroid can be decomposed into sequentially 4-connected pieces and some special 3-connected matroids (Beavers 2006 / Chen and Xiang, 2012).
  - Associated tree describes how to rebuild the original from the parts
  - The "gluing" operation is generalised parallel connection along a line

Decompositions revisited

- A 3-connected representable matroid can be decomposed into sequentially 4-connected pieces and some special 3-connected matroids (Beavers 2006 / Chen and Xiang, 2012).
  - Associated tree describes how to rebuild the original from the parts
  - The "gluing" operation is generalised parallel connection along a line

Constructing 3-trees

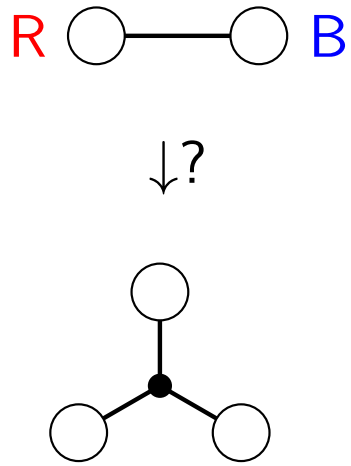
For an arbitrary 3-connected matroid, can we construct its 3-tree in polynomial time?

- The proof in OSW2004 doesn't lend itself to a polynomial-time algorithm
  - We can find, in polynomial time, a non-sequential 3-separation (R, B) if one exists
  - Not known if, in polynomial time, we can identify if (R, B) is a tight maximal flower or whether it can be refined

Constructing 3-trees

For an arbitrary 3-connected matroid, can we construct its 3-tree in polynomial time?

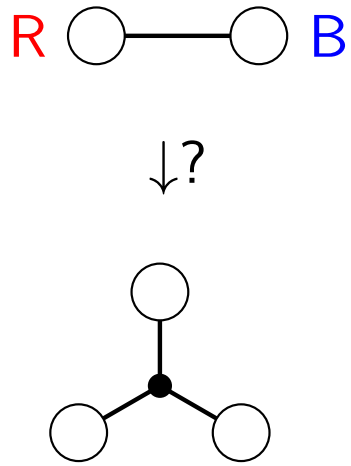
- The proof in OSW2004 doesn't lend itself to a polynomial-time algorithm
  - We can find, in polynomial time, a non-sequential 3-separation (R, B) if one exists
  - Not known if, in polynomial time, we can identify if (R, B) is a tight maximal flower or whether it can be refined



Constructing 3-trees

For an arbitrary 3-connected matroid, can we construct its 3-tree in polynomial time?

- The proof in OSW2004 doesn't lend itself to a polynomial-time algorithm
  - We can find, in polynomial time, a non-sequential 3-separation (R, B) if one exists
  - Not known if, in polynomial time, we can identify if (R, B) is a tight maximal flower or whether it can be refined



Finding non-sequential 3-separations

Theorem (Oxley and Semple, 2013)

A 3-tree for a 3-connected matroid can be constructed in polynomial time.

- Matroid specified by a rank oracle
- Need to find non-sequential 3-separations in polynomial time
  - The matroid intersection algorithm (Cunningham and Edmonds 1973) can be used to extend disjoint 3-element subsets X', Y' to a 3-separation (X, Y), if possible
  - Oxley and Semple showed that non-sequential 3-separations can be characterised by whether there are 3-element subsets X', Y' of X, Y (respectively) that are not contained in a maximal sequential set

Finding non-sequential 3-separations

Theorem (Oxley and Semple, 2013)

A 3-tree for a 3-connected matroid can be constructed in polynomial time.

- Matroid specified by a rank oracle
- Need to find non-sequential 3-separations in polynomial time
  - The matroid intersection algorithm (Cunningham and Edmonds 1973) can be used to extend disjoint 3-element subsets X', Y' to a 3-separation (X, Y), if possible
  - Oxley and Semple showed that non-sequential 3-separations can be characterised by whether there are 3-element subsets X', Y' of X, Y (respectively) that are not contained in a maximal sequential set

The 3-Tree algorithm

Theorem (Oxley and Semple, 2013)

A 3-tree for a 3-connected matroid can be constructed in polynomial time.

Outline of the 3-Tree algorithm:

1. ForwardSweep: Find the non-sequential 3-separations "in one direction" (a maximal 3-path)
2. BackwardSweep: Starting at the "unrooted" end, discover flower structure
3. While there is an unmarked bag B in the resulting tree
   - Fixing $E(M) - \pi(B)$ (at "rooted" end), call ForwardSweep, then BackwardSweep, to refine the structure of B, update the tree accordingly, then mark B

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)

$(X_1, X_2 \cup Y)$

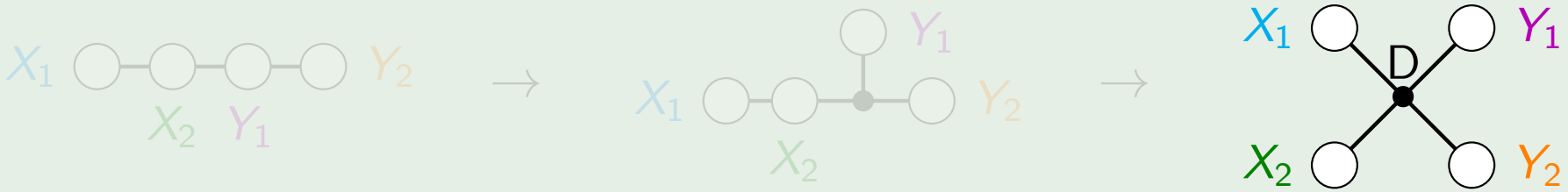
$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

BackwardSweep:

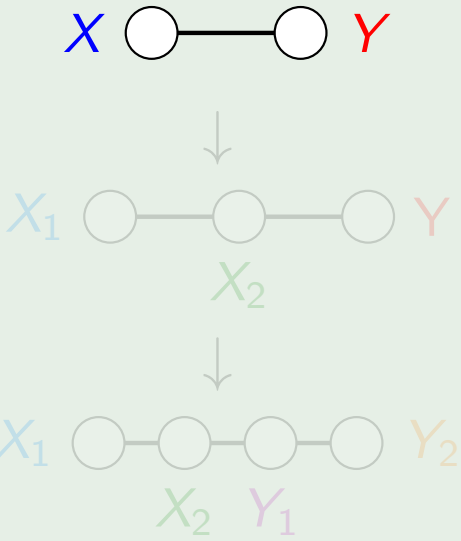


The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

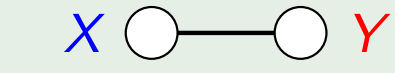
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

$(X \cup Y_1, Y_2)$

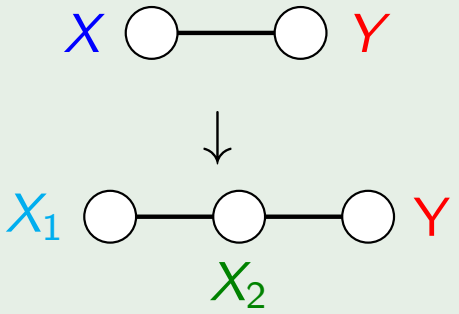
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

(X, Y)

(X_1, X_2, Y)

$(X \cup Y_1, Y_2)$

(X_1, X_2, Y_1, Y_2)

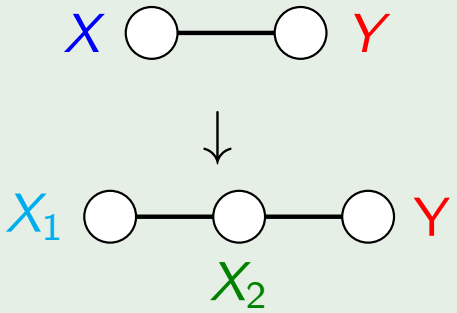
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

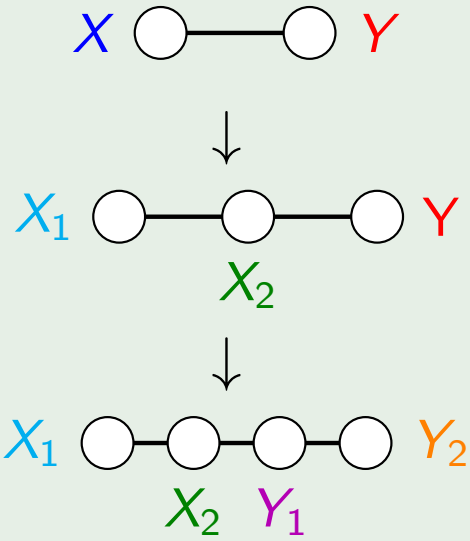
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

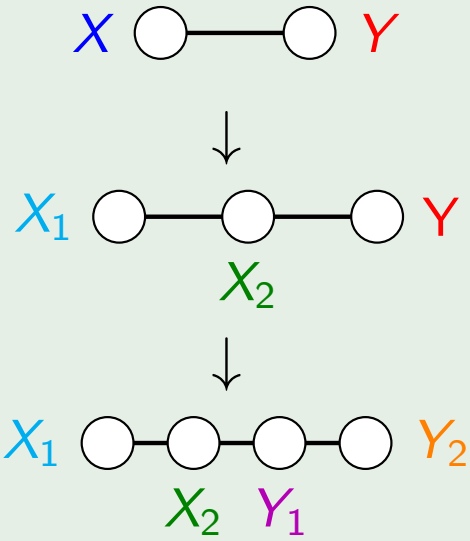
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

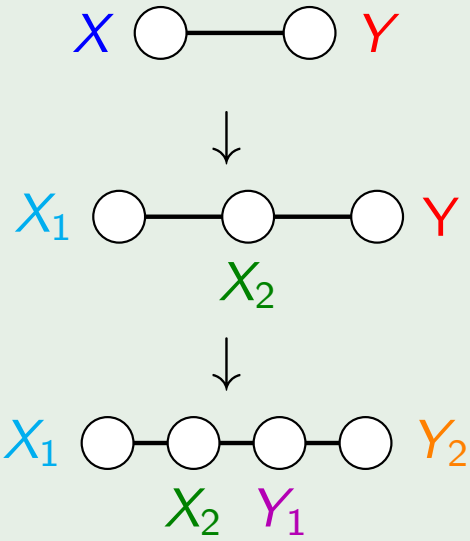
BackwardSweep:

The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

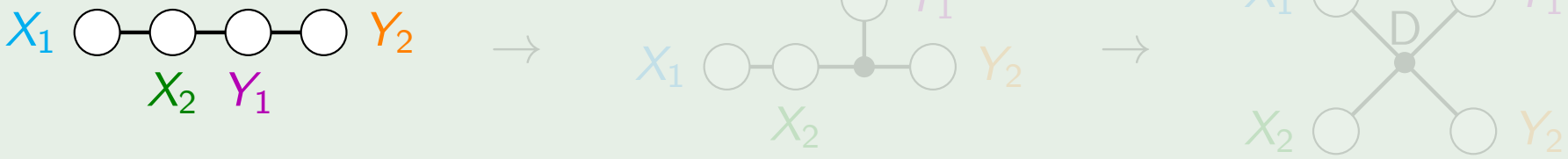
$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

BackwardSweep:

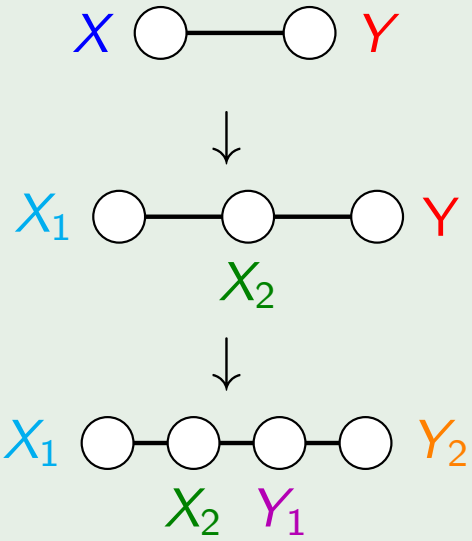


The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

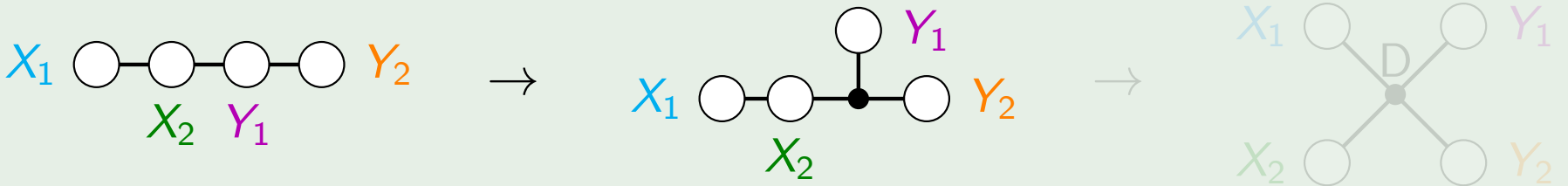
$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

BackwardSweep:

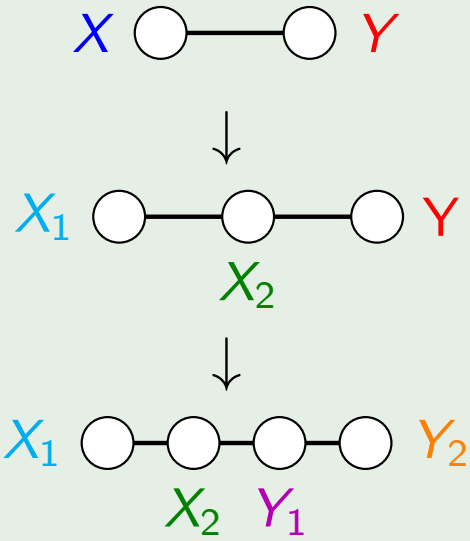


The 3-tree algorithm: an example

Example

ForwardSweep:

(X, Y)



$(X_1, X_2 \cup Y)$

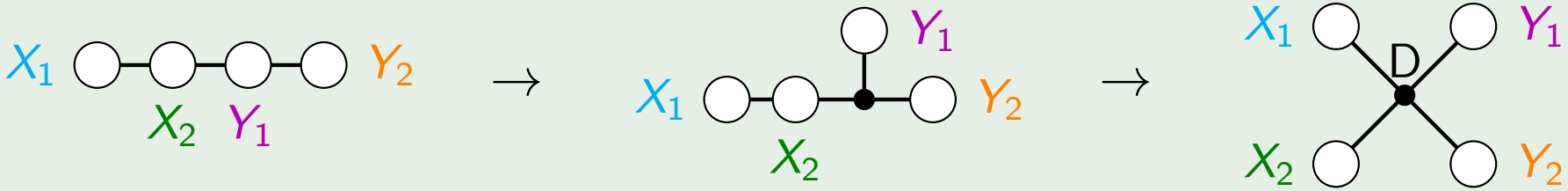
$(X \cup Y_1, Y_2)$

(X, Y)

(X_1, X_2, Y)

(X_1, X_2, Y_1, Y_2)

BackwardSweep:



4-trees

Theorem (Aikin and Oxley, 2012)

For every 4-connected matroid with $|E(M)| \geq 17$, there is a tree that displays, up to a natural equivalence, all the non-sequential 4-separations.

- Called a 4-tree
- For 4-separations, it is natural to allow up to two elements to be moved across the 4-separation
- Define "full closure" operator (and non-sequential 4-separations) accordingly
- Whereas a tight 3-flower has no unnecessary petals, for a 4-flower to have no unnecessary petals it also needs to be irredundant

4-trees

Theorem (Aikin and Oxley, 2012)

For every 4-connected matroid with $|E(M)| \geq 17$, there is a tree that displays, up to a natural equivalence, all the non-sequential 4-separations.

- Called a 4-tree
- For 4-separations, it is natural to allow up to two elements to be moved across the 4-separation
- Define "full closure" operator (and non-sequential 4-separations) accordingly
- Whereas a tight 3-flower has no unnecessary petals, for a 4-flower to have no unnecessary petals it also needs to be irredundant

4-trees

Theorem (Aikin and Oxley, 2012)

For every 4-connected matroid with $|E(M)| \geq 17$, there is a tree that displays, up to a natural equivalence, all the non-sequential 4-separations.

- Called a 4-tree
- For 4-separations, it is natural to allow up to two elements to be moved across the 4-separation
- Define "full closure" operator (and non-sequential 4-separations) accordingly
- Whereas a tight 3-flower has no unnecessary petals, for a 4-flower to have no unnecessary petals it also needs to be irredundant

k-trees

Theorem (Clark and Whittle, 2013)

For any tangle of order k in a connectivity system satisfying a "robustness" condition, there is a tree that displays the non-trivial k-separations with respect to the tangle.

Corollary

For any k-connected matroid M with $|E(M)| \geq 8k - 15$, there is a tree that displays, up to a natural equivalence, all the non-sequential k-separations.

- A k-tree

k-trees

Theorem (Clark and Whittle, 2013)

For any tangle of order k in a connectivity system satisfying a "robustness" condition, there is a tree that displays the non-trivial k-separations with respect to the tangle.

Corollary

For any k-connected matroid M with $|E(M)| \geq 8k - 15$, there is a tree that displays, up to a natural equivalence, all the non-sequential k-separations.

- A k-tree

k-equivalence

Corollary (Clark and Whittle, 2013)

For any k-connected matroid M, there is a tree that displays, up to k-equivalence, all the k-separations that are not k-sequential.

Let $(X, E - X)$ be a k-separation.

- $(Y_i)_{i=1}^m$ is a partial k-sequence for X if $|Y_j| \leq k-2$ and $X \cup Y_1 \cup \cdots \cup Y_j$ is k-separating for each $j \in \{1, 2, \ldots, m\}$.
- When $(Y_i)_{i=1}^m$ is maximal, $\mathrm{fcl}_k(X) = X \cup Y_1 \cup \cdots \cup Y_i$
- (X, Y) is k-sequential if $\mathrm{fcl}_k(X) = E(M)$ or $\mathrm{fcl}_k(Y) = E(M)$
- (X, Y) is k-equivalent to (A, B) if $\{\mathrm{fcl}_k(X), \mathrm{fcl}_k(Y)\} = \{\mathrm{fcl}_k(A), \mathrm{fcl}_k(B)\}$

Constructing k-trees

Can we construct a k-tree in polynomial time?

Theorem (B, Semple, 2015)

A k-tree for a k-connected matroid can be constructed in polynomial time.

- First, need to be able find non-sequential k-separations in polynomial time

Lemma (B, Semple, 2015)

A k-separation (U, V) in a k-connected matroid is not k-sequential iff there are k-element sets $U' \subseteq U$ and $V' \subseteq V$ such that neither U' nor V' is contained in a maximal k-sequential set.

Constructing k-trees

Can we construct a k-tree in polynomial time?

Theorem (B, Semple, 2015)

A k-tree for a k-connected matroid can be constructed in polynomial time.

- First, need to be able find non-sequential k-separations in polynomial time

Lemma (B, Semple, 2015)

A k-separation (U, V) in a k-connected matroid is not k-sequential iff there are k-element sets $U' \subseteq U$ and $V' \subseteq V$ such that neither U' nor V' is contained in a maximal k-sequential set.

Constructing k-trees

Can we construct a k-tree in polynomial time?

Theorem (B, Semple, 2015)

A k-tree for a k-connected matroid can be constructed in polynomial time.

- First, need to be able find non-sequential k-separations in polynomial time

Lemma (B, Semple, 2015)

A k-separation (U, V) in a k-connected matroid is not k-sequential iff there are k-element sets $U' \subseteq U$ and $V' \subseteq V$ such that neither U' nor V' is contained in a maximal k-sequential set.

The k-Tree algorithm

Theorem (B, Semple, 2015)

A k-tree for a k-connected matroid can be constructed in polynomial time.

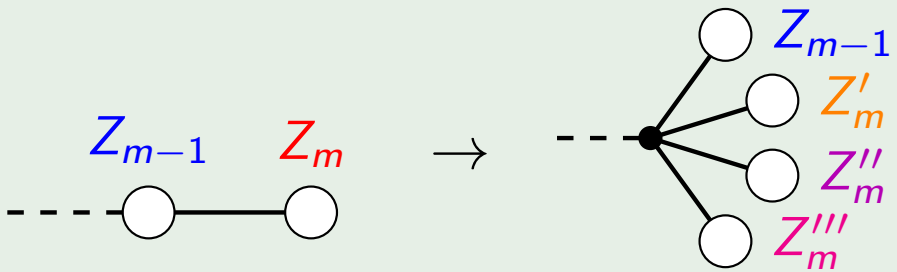
The k-Tree algorithm:

- Similar process to 3-Tree
  1. Find a non-sequential k-separation
  2. ForwardSweep
  3. BackwardSweep
  4. Repeat while there is an unmarked bag
- ForwardSweep process also very similar (replace 3's with k's)
- BackwardSweep is rather more complicated

BackwardSweep complication 1: k-sequential petals

- An end part of a 3-path can break into at most 2 sequential petals
- An end part of a 4-path can break into at most 3 sequential petals

Example



- Thankfully, this trend doesn't continue

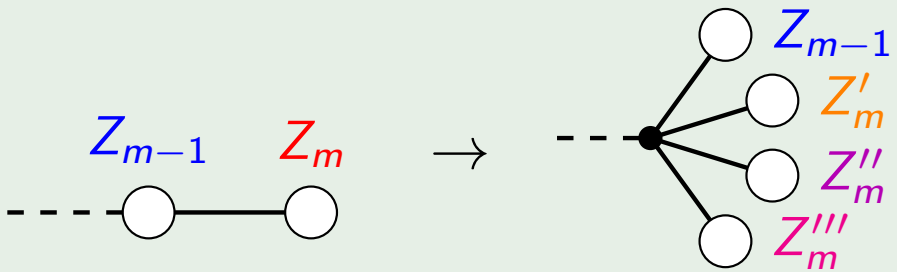
Lemma (B, Semple, 2015)

No union of three consecutive tight petals in a k-flower is k-sequential.

BackwardSweep complication 1: k-sequential petals

- An end part of a 3-path can break into at most 2 sequential petals
- An end part of a 4-path can break into at most 3 sequential petals

Example



- Thankfully, this trend doesn't continue

Lemma (B, Semple, 2015)

No union of three consecutive tight petals in a k-flower is k-sequential.

BackwardSweep complication 2: redundant flowers

- Ensuring the algorithm produces irredundant flowers
  - A k-anemone $\Phi = (X_1, X_2, \ldots, X_n)$ is irredundant if, for all distinct $i, j \in \{1, 2, \ldots, n\}$, there is a non-sequential k-separation (Y, Z) displayed by Φ for which $X_i \subseteq Y$ and $X_j \subseteq Z$
  - Similar for a k-daisy, but with consecutive i, j.
  - A tight 3-flower is irredundant, but a tight k-flower (for $k \geq 4$) may not be
  - A tight irredundant k-flower has no unnecessary petals
- Extra checks, that a certain k-separation exists, are required when refining end petals

BackwardSweep complication 2: redundant flowers

- Ensuring the algorithm produces irredundant flowers
  - A k-anemone $\Phi = (X_1, X_2, \ldots, X_n)$ is irredundant if, for all distinct $i, j \in \{1, 2, \ldots, n\}$, there is a non-sequential k-separation (Y, Z) displayed by Φ for which $X_i \subseteq Y$ and $X_j \subseteq Z$
  - Similar for a k-daisy, but with consecutive i, j.
  - A tight 3-flower is irredundant, but a tight k-flower (for $k \geq 4$) may not be
  - A tight irredundant k-flower has no unnecessary petals
- Extra checks, that a certain k-separation exists, are required when refining end petals

Thanks for your attention