Constructing brambles

Mathieu Chapelle[1], Frédéric Mazoit[2], Ioan Todinca[1]

[1]LIFO – Université d'Orléans
[2]LaBRI – Université de Bordeaux I

JGA'09
Montpellier
5–6 novembre 2009

Tree-like decompositions

A popular technique used when dealing with $\mathcal{NP}$-hard problems is to decompose the input graph and then use *dynamic programming*.

Tree-like decompositions

A popular technique used when dealing with $\mathcal{NP}$-hard problems is to decompose the input graph and then use *dynamic programming*.
e.g.: *tree-width*, branch-width, rank-width, clique-width, ...

Tree-like decompositions

A popular technique used when dealing with $\mathcal{NP}$-hard problems is to decompose the input graph and then use *dynamic programming*.
e.g.: *tree-width*, branch-width, rank-width, clique-width, ...

All works in same flavor:

- decompose recursively the graph, and glue subparts in a kind of tree;
- the $*$-width is given by this decomposition;
- apply a bottom-up approach, and glue sub-solutions to obtain a global solution.

Tree-like decompositions

A popular technique used when dealing with $\mathcal{NP}$-hard problems is to decompose the input graph and then use *dynamic programming*.
e.g.: *tree-width*, branch-width, rank-width, clique-width, ...

All works in same flavor:

- decompose recursively the graph, and glue subparts in a kind of tree;
- the $*$-width is given by this decomposition;
- apply a bottom-up approach, and glue sub-solutions to obtain a global solution.

Lots of usual $\mathcal{NP}$-hard problems can be solved in polynomial or linear time when restricted to graphs with bounded $*$-widths.

Tree-width and bramble

How can we argue that the tree-width of a graph is at most k?

Tree-width and bramble

How can we argue that the tree-width of a graph is at most k?
It is sufficient to provide a tree-decomposition of width k of the graph.

Tree-width and bramble

How can we argue that the tree-width of a graph is at most k?
It is sufficient to provide a tree-decomposition of width k of the graph.

On the contrary, it is more tricky to argue that the tree-width of a graph is at least $k+1$.

Tree-width and bramble

How can we argue that the tree-width of a graph is at most k?
It is sufficient to provide a tree-decomposition of width k of the graph.

On the contrary, it is more tricky to argue that the tree-width of a graph is at least $k+1$.
This is given by a bramble (of order $k+2$), a combinatorial object which will act as a certificate that the tree-width of the graph can't be less than $k+1$.

Intuition

Cops-and-robber game for tree-width:

Intuition

Cops-and-robber game for tree-width:

- tree-decomposition of width at most k $\rightarrow$ winning strategy for $k+1$ cops;

Intuition

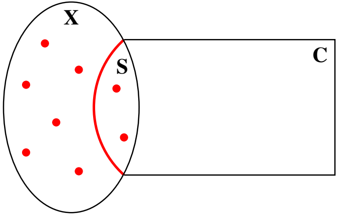
Cops-and-robber game for tree-width:

- tree-decomposition of width at most $k \to$ winning strategy for $k+1$ cops;
- if $k+1$ cops is not enough $\to$ winning strategy for the fugitive.

Intuition

Cops-and-robber game for tree-width:

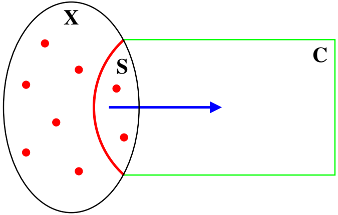
- tree-decomposition of width at most k $\rightarrow$ winning strategy for $k+1$ cops;
- if $k+1$ cops is not enough $\rightarrow$ winning strategy for the fugitive.



Intuition

Cops-and-robber game for tree-width:

- tree-decomposition of width at most $k \rightarrow$ winning strategy for $k+1$ cops;
- if $k+1$ cops is not enough $\rightarrow$ winning strategy for the fugitive.



Duality theorem for tree-width

Theorem ([Seymour, Thomas (92)])

A graph G has tree-width strictly larger than k if and only if G has a bramble of order $k+2$.

Duality theorem for tree-width

Theorem ([Seymour, Thomas (92)])

A graph G has tree-width strictly larger than k if and only if G has a bramble of order $k+2$.

This follow the natural intuition of cops-and-robber game: there can't be a winning strategy for both players (the k cops and the fugitive).

Duality theorem for tree-width

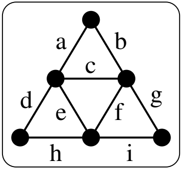
Theorem ([Seymour, Thomas (92)])

A graph G has tree-width strictly larger than k if and only if G has a bramble of order $k+2$.

This follow the natural intuition of cops-and-robber game: there can't be a winning strategy for both players (the k cops and the fugitive).

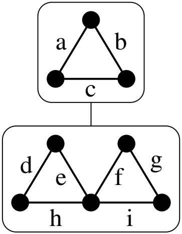
He we present the first (non-trivial) exact algorithm to construct an optimal bramble in time $\mathcal{O}(n^{k+4})$.

Partitioning trees



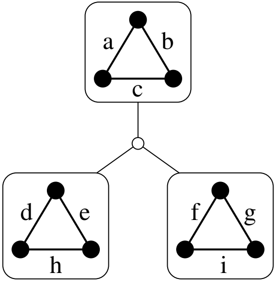
We start with a node containing every edges of the graph,

Partitioning trees



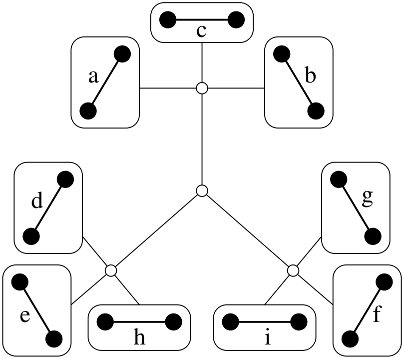
We start with a node containing every edges of the graph, and we *recursively decompose* it

Partitioning trees



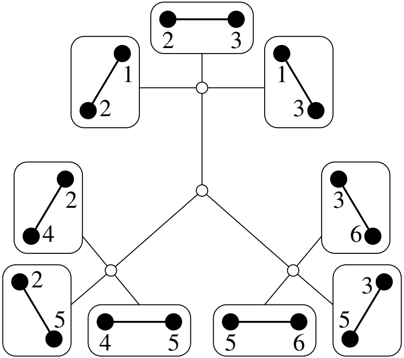
We start with a node containing every edges of the graph, and we *recursively decompose* it

Partitioning trees



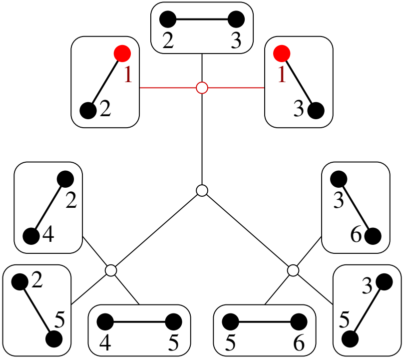
We start with a node containing every edges of the graph, and we *recursively decompose* it until we obtain a *partitioning tree* of the graph.

Partitioning trees



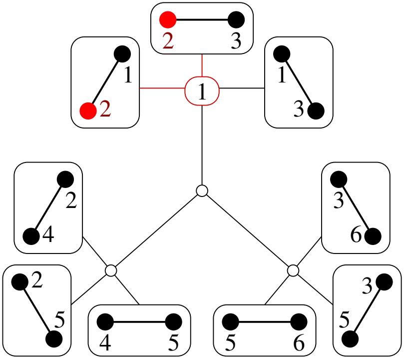
Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



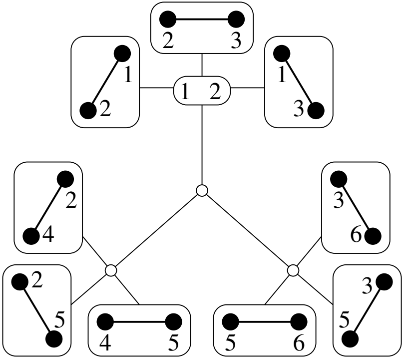
Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



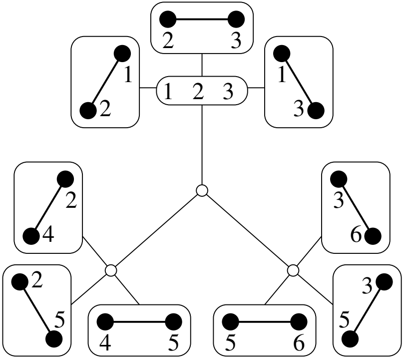
Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



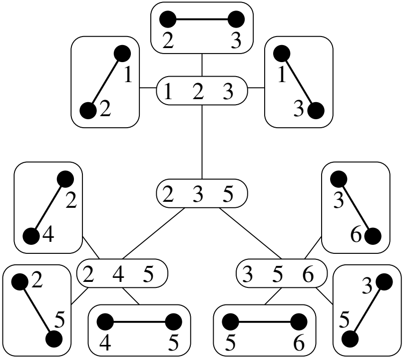
Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



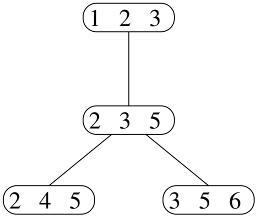
Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



Now, let us exhibit the border of each internal node, *i.e.* the vertices appearing in at least two leaves of the internal node.

Partitioning trees



We end up with a tree decomposition of the initial graph.
Tree-width $\leq k$ iff there exists a partitioning tree with internal bags $\leq k+1$.

$\mathcal{P}_k$-partitioning trees

$\mathcal{P}_k = \{\mu : |\delta(\mu)| \leq k\}$ is the family of partitions of E with borders size at most k.

$\mathcal{P}_k$-partitioning trees

$\mathcal{P}_k = \{\mu : |\delta(\mu)| \leq k\}$ is the family of partitions of E with borders size at most k.

Definition ($\mathcal{P}_k$-partitioning tree)

A $\mathcal{P}_k$*-partitioning tree* (T, τ) of G is a tree whose set of leaves is the set of edges of G, and $\forall v \in T, \mu_v \in \mathcal{P}_k$.

$\mathcal{P}_k$-partitioning trees

$\mathcal{P}_k = \{\mu : |\delta(\mu)| \leq k\}$ is the family of partitions of E with borders size at most k.

Definition ($\mathcal{P}_k$-partitioning tree)

A $\mathcal{P}_k$*-partitioning tree* (T, τ) of G is a tree whose set of leaves is the set of edges of G, and $\forall v \in T, \mu_v \in \mathcal{P}_k$.

Thus a partitioning tree is a recursive decomposition of the edge set E.

$\mathcal{P}_k$-partitioning trees

$\mathcal{P}_k = \{\mu : |\delta(\mu)| \leq k\}$ is the family of partitions of E with borders size at most k.

Definition ($\mathcal{P}_k$-partitioning tree)

A *$\mathcal{P}_k$-partitioning tree* (T, τ) of G is a tree whose set of leaves is the set of edges of G, and $\forall v \in T, \mu_v \in \mathcal{P}_k$.

Thus a partitioning tree is a recursive decomposition of the edge set E.

The partitions of $\mathcal{P}_k$ correspond to partial partitioning stars (*i.e.* trees with only one internal node).

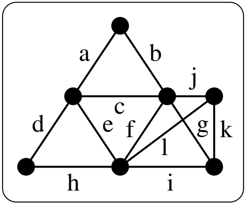
Recall that we deal with graphs of tree-width strictly greater than k.

Recall that we deal with graphs of tree-width strictly greater than k.

We use partial partitioning trees, which are just like partitioning trees, but whose leaves can contain arbitrary subsets of edges (not necessarily uniques), and where internal nodes are still of border at most k.

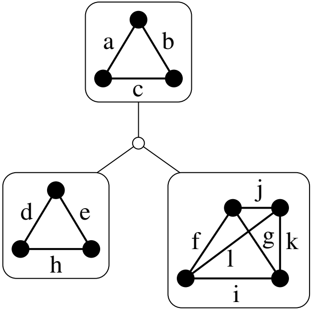
Recall that we deal with graphs of tree-width strictly greater than k.

We use partial partitioning trees, which are just like partitioning trees, but whose leaves can contain arbitrary subsets of edges (not necessarily uniques), and where internal nodes are still of border at most k.



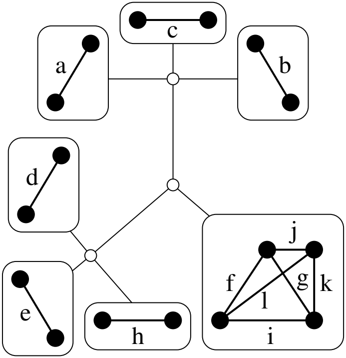
Recall that we deal with graphs of tree-width strictly greater than k.

We use partial partitioning trees, which are just like partitioning trees, but whose leaves can contain arbitrary subsets of edges (not necessarily uniques), and where internal nodes are still of border at most k.



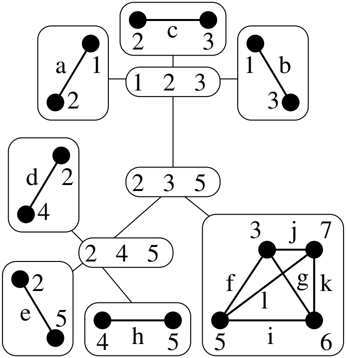
Recall that we deal with graphs of tree-width strictly greater than k.

We use partial partitioning trees, which are just like partitioning trees, but whose leaves can contain arbitrary subsets of edges (not necessarily uniques), and where internal nodes are still of border at most k.



Recall that we deal with graphs of tree-width strictly greater than k.

We use partial partitioning trees, which are just like partitioning trees, but whose leaves can contain arbitrary subsets of edges (not necessarily uniques), and where internal nodes are still of border at most k.



$\mathcal{P}$-bramble

Definition ($\mathcal{P}$-bramble)

A $\mathcal{P}$-bramble $\mathcal{B}$ of $G = (V, E)$ is a family of subsets of E containing a non-trivial part (*i.e.* not a single edge) of every partition $(E_1, \ldots, E_p) \in \mathcal{P}$, and such that they are pairwise intersecting.*

*In the usual definition of a bramble, the elements are subsets of V and need to be pairwise touching.

$\mathcal{P}$-bramble

Definition ($\mathcal{P}$-bramble)

A $\mathcal{P}$-bramble $\mathcal{B}$ of $G = (V, E)$ is a family of subsets of E containing a non-trivial part (*i.e.* not a single edge) of every partition $(E_1, \ldots, E_p) \in \mathcal{P}$, and such that they are pairwise intersecting.*

Note that computing the order of a given bramble is $\mathcal{NP}$-hard (it is the minimum cardinality of a hitting set).

*In the usual definition of a bramble, the elements are subsets of V and need to be pairwise touching.

More on the duality theorem

Suppose that tree-width is greater than k.

More on the duality theorem

Suppose that tree-width is greater than k.

Let $\mathcal{B}$ be a set containing a (non-trivial) part of every partition in $\mathcal{P}^{\uparrow}$, upward-closed, and minimal.

More on the duality theorem

Suppose that tree-width is greater than k.

Let $\mathcal{B}$ be a set containing a (non-trivial) part of every partition in $\mathcal{P}^{\uparrow}$, upward-closed, and minimal.

Theorem ([Lyaudet, Mazoit, Thomassé (09)])

$\mathcal{B}$ is a $\mathcal{P}^{\uparrow}$-bramble.

More on the duality theorem

Suppose that tree-width is greater than k.

Let $\mathcal{B}$ be a set containing a (non-trivial) part of every partition in $\mathcal{P}^{\uparrow}$, upward-closed, and minimal.

Theorem ([Lyaudet, Mazoit, Thomassé (09)])

$\mathcal{B}$ is a $\mathcal{P}^{\uparrow}$-bramble.

Our algorithm uses this result to construct a $\mathcal{P}^{\uparrow}$-bramble.

The problem

Tree-decompositions and brambles revisited

The algorithm

Conclusion and perspectives

First algorithm

`Bramble`$(\mathcal{P}_k)$

$\mathcal{B} \leftarrow \mathit{Flaps}(\mathcal{P}_k^{\uparrow})$

foreach $X \in \mathcal{B}$ taken by increasing size **do**

if there is no $\mu \in \mathcal{P}_k^{\uparrow}$ with $\mathit{Flaps}(\mu) \cap \mathcal{B} = \{X\}$ **and**

X strictly contains no $Y \in \mathcal{B}$

then $\mathcal{B} \leftarrow \mathcal{B} \setminus \{X\}$

return $\mathcal{B}$

First algorithm

`Bramble`$(\mathcal{P}_k)$

$\mathcal{B} \leftarrow \mathit{Flaps}(\mathcal{P}_k^{\uparrow})$

foreach $X \in \mathcal{B}$ taken by increasing size **do**

if there is no $\mu \in \mathcal{P}_k^{\uparrow}$ with $\mathit{Flaps}(\mu) \cap \mathcal{B} = \{X\}$ **and**

X strictly contains no $Y \in \mathcal{B}$

then $\mathcal{B} \leftarrow \mathcal{B} \setminus \{X\}$

return $\mathcal{B}$

Problem

The size of $\mathcal{P}_k^{\uparrow}$ is exponential in $|\mathcal{P}_k|$.

First algorithm

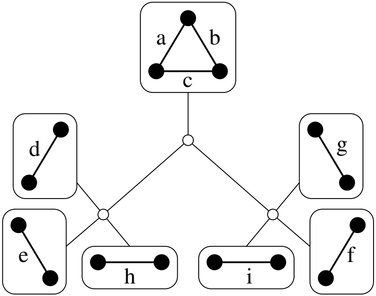
```
Bramble(P_k)
    B ← Flaps(P_k)
    foreach X ∈ B taken by increasing size do
        if there is no μ ∈ P_k with Flaps(μ) ∩ B = {X} and
        X strictly contains no Y ∈ B
        then B ← B \ {X}
    return B
```

Problem

The size of $\mathcal{P}_k^{\uparrow}$ is exponential in $|\mathcal{P}_k|$.

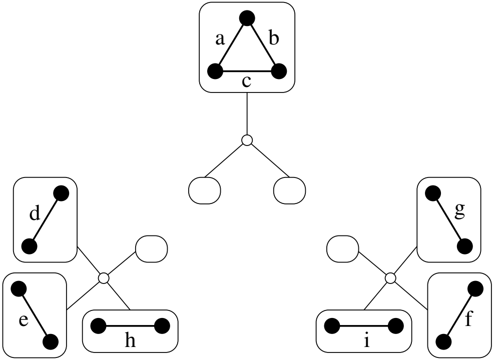
Anticipate forced decisions

Mark the forbidden flaps and the forced flaps. A flap is forced if it is the unique non-trivial leaf of some partial partioning tree.



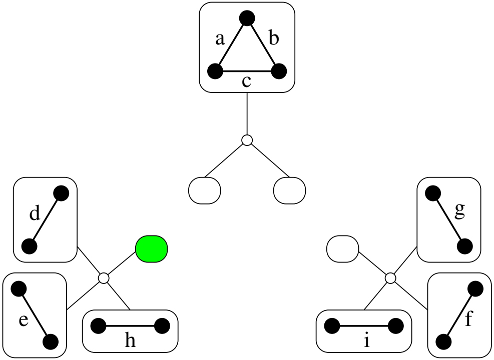
Anticipate forced decisions

Mark the forbidden flaps and the forced flaps. A flap is forced if it is the unique non-trivial leaf of some partial partioning tree.



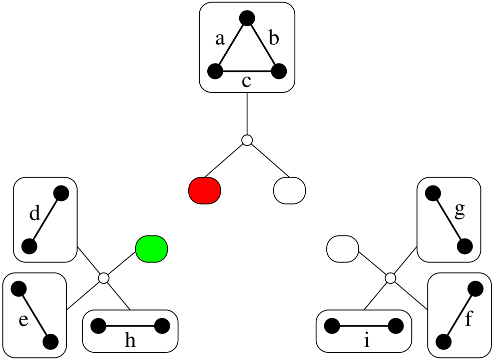
Anticipate forced decisions

Mark the forbidden flaps and the forced flaps. A flap is forced if it is the unique non-trivial leaf of some partial partioning tree.



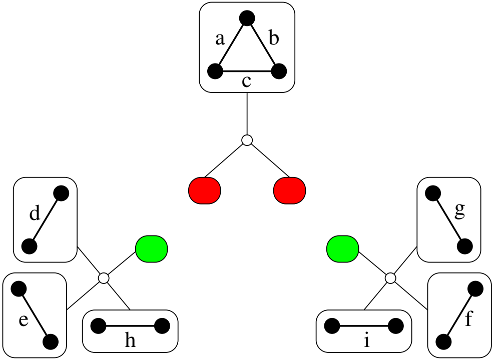
Anticipate forced decisions

Mark the forbidden flaps and the forced flaps. A flap is forced if it is the unique non-trivial leaf of some partial partioning tree.



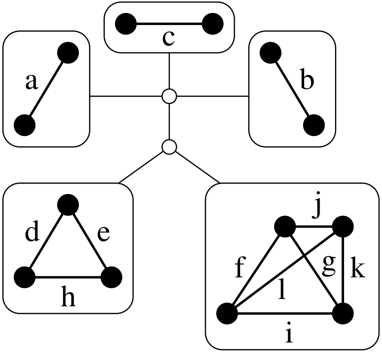
Anticipate forced decisions

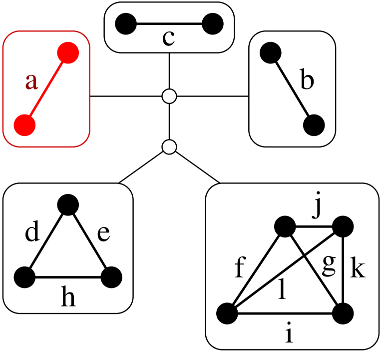
Mark the forbidden flaps and the forced flaps. A flap is forced if it is the unique non-trivial leaf of some partial partioning tree.

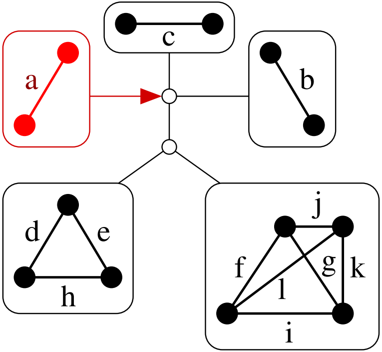


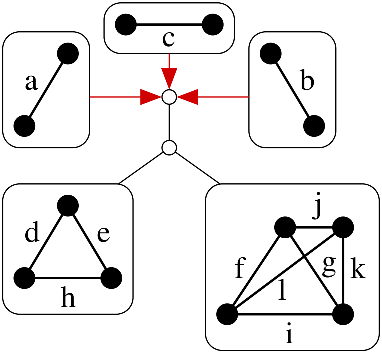
Algorithm with marking process

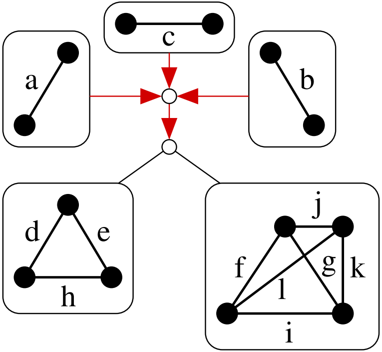
```
Bramble(𝒫_k)
    ℬ ← Flaps(𝒫_k); UpdateMarks
    foreach X ∈ ℬ taken by increasing size do
        if X is not marked as forced and
        X strictly contains no Y ∈ ℬ
        then ℬ ← ℬ \ {X}; UpdateMarks
    return ℬ

UpdateMarks
    while there exists μ ∈ 𝒫_k with Flaps(μ) ∩ ℬ = {X} do
        Mark X as forced;
```

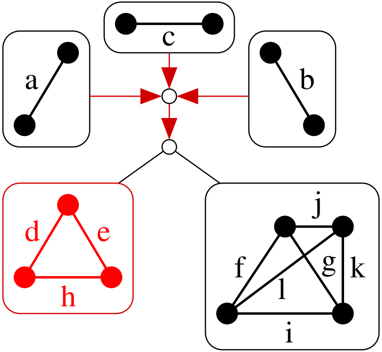

c
a
b
d
e
h
j
f
g
k
l
i

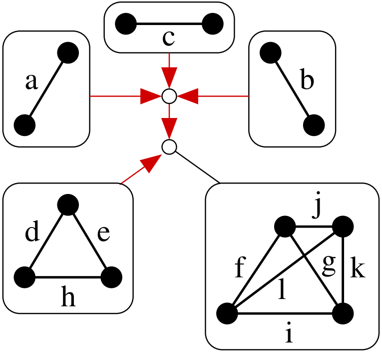

a
c
b
d
e
h
j
f
g
k
l
i

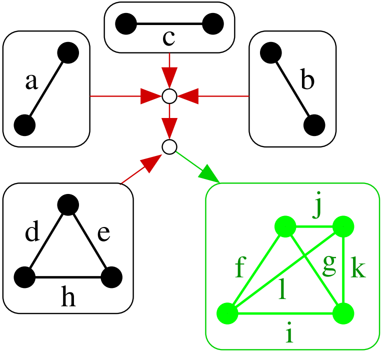

a
c
b
d
e
h
j
f
g
k
l
i


c
a
b
d
e
h
j
f
g
k
l
i


c
a
b
d
e
h
j
f
g
k
l
i


c
a
b
j
d
e
f
g
k
l
h
i


c
a
b
j
d
e
f
g
k
l
h
i


c
a
b
d
e
h
j
f
g
k
l
i

Algorithm with marking process

`Bramble`$(\mathcal{P}_k)$

$\mathcal{B} \leftarrow$ *Flaps*$(\mathcal{P}_k)$; `UpdateMarks`

foreach $X \in \mathcal{B}$ taken by increasing size **do**

if X is not marked as *forced* **and**

X strictly contains no $Y \in \mathcal{B}$

then $\mathcal{B} \leftarrow \mathcal{B} \setminus \{X\}$; `UpdateMarks`

return $\mathcal{B}$

`UpdateMarks`

while there exists $\mu \in \mathcal{P}_k$ with *Flaps*$(\mu) \cap \mathcal{B} = \{X\}$ **do**

Mark X as *forced*;

Theorem

The time complexity of `Bramble` *is polynomial in* $|\mathcal{P}_k|$.

Algorithm with marking process

```
Bramble(𝒫_k)
    ℬ ← Flaps(𝒫_k); UpdateMarks
    foreach X ∈ ℬ taken by increasing size do
        if X is not marked as forced and
        X strictly contains no Y ∈ ℬ
        then ℬ ← ℬ \ {X}; UpdateMarks
    return ℬ

UpdateMarks
    while there exists μ ∈ 𝒫_k with Flaps(μ) ∩ ℬ = {X} do
        Mark X as forced;
```

Theorem

The time complexity of `Bramble` *is* $\mathcal{O}(|\mathcal{P}_k| n^4)$.

Algorithm with marking process

`Bramble`$(\mathcal{P}_k)$
- $\mathcal{B} \leftarrow Flaps(\mathcal{P}_k)$; `UpdateMarks`
- **foreach** $X \in \mathcal{B}$ taken by increasing size **do**
  - **if** X is not marked as *forced* **and**
  - X strictly contains no $Y \in \mathcal{B}$
  - **then** $\mathcal{B} \leftarrow \mathcal{B} \setminus \{X\}$; `UpdateMarks`
- **return** $\mathcal{B}$

`UpdateMarks`
- **while** there exists $\mu \in \mathcal{P}_k$ with $Flaps(\mu) \cap \mathcal{B} = \{X\}$ **do**
  - Mark X as *forced*;

Theorem

The time complexity of `Bramble` *is* $\mathcal{O}(n^{k+4})$.

(using data structure as in [Arnborg, Corneil, Proskurowski (87)])

The problem

Tree-decompositions and brambles revisited

The algorithm

Conclusion and perspectives

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

- There always exists a bramble of size $f(k)$ (by kernelization), but $f(k)$ can be exponential ([Grohe, Marx (09)]).

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

- There always exists a bramble of size $f(k)$ (by kernelization), but $f(k)$ can be exponential ([Grohe, Marx (09)]).
- Computing the order of a bramble is $\mathcal{NP}$-hard (hitting set).

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

- There always exists a bramble of size $f(k)$ (by kernelization), but $f(k)$ can be exponential ([Grohe, Marx (09)]).
- Computing the order of a bramble is $\mathcal{NP}$-hard (hitting set).

Perspectives:

- Better bramble (in size) or better time complexity?

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

- There always exists a bramble of size $f(k)$ (by kernelization), but $f(k)$ can be exponential ([Grohe, Marx (09)]).
- Computing the order of a bramble is $\mathcal{NP}$-hard (hitting set).

Perspectives:

- Better bramble (in size) or better time complexity?
- Can we extend this approach to other tree-like decompositions?

Theorem

There is an algorithm computing in time $\mathcal{O}(n^{k+4})$, either a tree-decomposition of width at most k, or a bramble of order $k+2$.

- There always exists a bramble of size $f(k)$ (by kernelization), but $f(k)$ can be exponential ([Grohe, Marx (09)]).
- Computing the order of a bramble is $\mathcal{NP}$-hard (hitting set).

Perspectives:

- Better bramble (in size) or better time complexity?
- Can we extend this approach to other tree-like decompositions?
- Approximate brambles (see *e.g.* [Kreutzer, Tazari (SODA'10)])?