4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE
http://www.univ-orleans.fr/lifo

Rapport de Recherche

Constructing brambles

Mathieu Chapelle
LIFO, Université d'Orléans

Frédéric Mazoit
LaBRI, Université de Bordeaux

Ioan Todinca
LIFO, Université d'Orléans

Rapport n^o **RR-2009-04**

Constructing brambles

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

[1] Université d'Orléans, LIFO, BP-6759, F-45067 Orléans Cedex 2, France
{mathieu.chapelle,ioan.todinca}@univ-orleans.fr
[2] Université de Bordeaux, LaBRI, F-33405 Talence Cedex, France
Frederic.Mazoit@labri.fr

Abstract. Given an arbitrary graph G and a number k, it is well-known by a result of Seymour and Thomas [20] that G has treewidth strictly larger than k if and only if it has a bramble of order $k+2$. Brambles are used in combinatorics as certificates proving that the treewidth of a graph is large. From an algorithmic point of view there are several algorithms computing tree-decompositions of G of width at most k, if such decompositions exist and the running time is polynomial for constant k. Nevertheless, when the treewidth of the input graph is larger than k, to our knowledge there is no algorithm constructing a bramble of order $k+2$. We give here such an algorithm, running in $O(n^{k+4})$ time. Moreover, for classes of graphs with polynomial number of minimal separators, we define a notion of *compact brambles* and show how to compute compact brambles of order $k+2$ in polynomial time, not depending on k.

1 Introduction

Motivation. Treewidth is one of the most extensively studied graph parameters, mainly because many classical NP-hard optimisation problems become polynomial and even linear when restricted to graphs of bounded treewidth. In many applications of treewidth, one needs to compute a tree-decomposition of small width of the input graph. Although determining the treewidth of arbitrary graphs is NP-hard [2], for small values of k one can decide quite efficiently if the treewidth of the input graph is at most k. The first result of this flavour is a rather natural and simple algorithm due to Arnborg, Corneil and Proskurowski [2] working in $O(n^{k+2})$ time (as usual n denotes the number of vertices of the input graph and m denotes the number of its edges). Using much more sophisticated techniques, Bodlaender [5] solves the same problem by an algorithm running in $O(f(k)n)$ time, where f is an exponential function. The treewidth problem has been also studied for graph classes, and it was shown [10, 11] that for any class of graphs with polynomial number of minimal separators, the treewidth can be computed in polynomial time. In particular the problem is polynomial for several natural graph classes e.g. circle graphs, circular arc graphs or weakly chordal graphs. All cited algorithms are able to compute optimal tree-decompositions of the input graph.

In the last years, tree decompositions have been used for practicle applications which encouraged the developement of heuristic methods for tree-decompositions of small width [3, 12] and, in order to validate the quality of the decompositions, several authors also developed algorithms finding lower bounds for treewidth [9, 8, 7, 12].

Indeed, one of the main difficulties with treewidth is that we do not have simple certificates for large treewidth. We can easily argue that the treewidth of a graph G is at most k: it is sufficient to provide a tree-decomposition of width at most k, and one can check in linear time that the decomposition is correct. On the other hand, how to argue that the treewidth of G is (strictly) larger than k? For this purpose, Seymour and Thomas introduced the notion of *brambles*, defined below. The goal of this article is to give algorithms for computing brambles.

Some definitions. A *tree decomposition* of a graph $G = (V, E)$ is a tree TD such that each node of TD is a bag (vertex subset of G) and satisfies the following properties: (1) each vertex of G appears in at least one bag, (2) for each edge of G there is a bag containing both endpoints of the edge and (3) for any vertex x of G, the bags containing x form a connected subtree of TD. The *width* of the decomposition is the size of its largest bags, minus one. The *treewidth* of G is the minimum width over all tree decompositions of G.

Without loss of generality, we restrict to tree decompositions such that there is no bag contained into another. We say that a tree decomposition TD is finer than another tree decomposition TD' if each bag of TD is contained in some bag of TD'. Clearly, optimal tree decompositions are among minimal ones, with respect to the refining relation. A set of vertices of G is a *potential maximal clique* if it is the bag of some minimal tree decomposition. Potential maximal cliques of a graph are incomparable with respect to inclusion [10]. More important, the number of potential maximal cliques is polynomially bounded in the number of minimal separators of the graph [11], which implies that for several graph classes (circle, circular arc, weakly chordal graphs...) the number of potential maximal cliques is polynomially bounded in the size of the graph.

A *bramble* of order k of $G = (V, E)$ is a function β mapping each vertex subset X of size at most $k-1$ to a connected component $\beta(X)$ of $G - X$. We require that, for any subsets X, Y of V of size at most $k-1$, the components $\beta(X)$ and $\beta(Y)$ touch, i.e. they have a common vertex or an edge between the two.[3]

Theorem 1 (Treewidth-bramble duality, [20]). *A graph G is of treewidth strictly larger than k if and only if it has a bramble of order $k+2$.*

Very roughly, if a graph has treewidth larger than k, for each possible bag X of size at most $k+1$ the bramble points toward a component of $G-X$ which cannot be decomposed by a tree-decomposition of width at most k (restricted to $X \cup \beta(X)$) using X as a bag. Treewidth can be also stated in terms of a cops-and-robber game, and tree decompositions of width k correspond to winning strategies for $k+1$ cops, while brambles of order $k+2$ correspond to escaping strategies for the robber, if the number of cops is at most $k+1$ (see e.g. [6]).

Since optimal tree decompositions can be obtained using only potential maximal cliques as bags, we introduce now *compact brambles* as follows. Instead of associating to any set X of size less than k a component $\beta(X)$ of $G - X$, we only do this for the sets which are also potential maximal cliques. It is not hard to prove that, in Theorem 1, we can replace brambles by compact brambles. Indeed if the treewidth of G is larger than k then there is a bramble of order $k+2$, which can be restricted to a compact bramble of the same order. Conversely, suppose that there exists a compact bramble of order $k+2$ and a tree decomposition of width at most k. Consider a minimal, optimal tree decomposition of G. According to [20], there is a bag X of this decomposition intersecting each set $\beta(Y)$ of the bramble. By our choice X is a potential maximal clique of size at most $k+1$, thus X intersects $\beta(X)$ – a contradiction.

Our results. After the seminal paper of Seymour and Thomas proving Theorem 1, there were several results, either with shorter and simpler proofs [4] or for proving other duality theorems of similar flavour, concerning different types of tree-like decompositions, e. g. branch-decompositions, path-decompositions or rank-decompositions (see [1] for a survey). Unified

[3] Here we use one of the equivalent definitions of brambles, see e.g. [6] and the Conclusion for further discussions.

versions of these results have been given recently in [1] and [16]. We point out that all these proofs are purely combinatorial: even for a small constant k, it is not clear at all how to use these proofs for constructing brambles on polynomial time (e.g. the proofs of [20, 4] start by considering the set of all connected subgraphs of the input graph). Nevertheless the recent construction of [16] gives a simpler information on the structure of a bramble (see Theorem 4). Based on their framework which we extend, we build up the following algorithmic result:

Theorem 2. *There is an algorithm that, given a graph G and a number k, computes either a tree decomposition of G of width at most k, or a bramble of order $k+2$, in $O(n^{k+4})$ time.*

There is an algorithm that, given a graph G, computes a tree-decomposition of width at most k or a compact bramble of order $k+2$ in $O(n^4r^2)$ time, where r is the number of minimal separators of the graph.

These brambles can be used as certificates that allow to check, by a simple algorithm, that a graph has treewidth larger than k. The certificates are of polynomial size for fixed k, and the compact brambles are of polynomial size for graph classes with a polynomial number of minimal separators, like circle, circular-arc or weakly chordal graphs (even for large k). Also, in the area of graph searching (see [15] for a survey), in which it is very common to give optimal strategies for the cops, our construction of brambles provides a simple escape strategy for the robber.

The paper is organized as follows. In Section 2 we introduce an extended version of the framework of [16] and their version of the duality theorem. In Section 3 we give the algorithm constructing the brambles. In the Conclusion we discuss extentions of these results and further research.

2 Treewidth, partitioning trees and the generalized duality theorem

For our purpose, it is more convenient to view tree decompositions as recursive decompositions of the edge set of the graph, thus we rather use the notion of partitioning trees. The notion is very similar to branch-decompositions [19] except that in our case internal nodes have arbitrary degree; in branch-decompositions, internal nodes are of degree three.

Definition 1 (partitioning trees). *A* partitioning tree *of a graph $G = (V, E)$ is a pair (T, τ) where T is a tree and τ is a one-to-one mapping of the edges of G on the leaves of T.*

Given an internal node i of T, let $\mu(i)$ be the partition of E where each part corresponds to the edges of G mapped on the leaves of a subtree of T obtained by removing node i.

We denote by $\delta(\mu(i))$ the border *of the partition, i.e. the set of vertices of G appearing in at least two parts of $\mu(i)$; $\delta(\mu(i))$ is also called the* bag *associated to node i. Let $width(T, \tau)$ be the $\max|\delta(\mu(i))| - 1$, over all internal nodes i of T. The treewidth of G is the minimum width over all partitioning trees of G.*

One can easily transform a partitioning tree (T, τ) into a tree decomposition such that the bags of the tree-decompositions are exactly the sets $\delta(\mu(i))$. Indeed consider the tree decomposition with the same tree, associate to each internal node i the bag $\delta(\mu(i))$ and to each leaf j the bag $\{x_j, y_j\}$ where x_jy_j is the edge of G mapped on the leaf j. It is a matter of routine to check that this tree decomposition satisfies all conditions of tree decompositions.

Conversely, consider a tree decomposition TD of G, we describe a partitioning tree (T, τ) obtained from TD without increasing the width. Initially T is a copy TD. For each edge e of

G, add a new leaf mapped to this edge, and make it adjacent to one of the nodes of T such that the corresponding bag contains both endpoints of the edge. Eventually, we recursively remove the leaves of T which correspond to nodes of TD (thus there is no edge of G mapped on these leaves). In the end we obtain a partitioning tree of G, and again is not hard to check that (see e.g. [17]) for each internal node i of T, we have that $\delta(\mu(i))$ is contained in the bag $X(i)$ corresponding to node i in TD.

Consequently, the treewidth of G is indeed the minimum width over all partitioning trees of G.

Given a graph which is not necessarily of treewidth at most k, we want to capture the "best decompositions" one can obtain with bags of size at most k. For this purpose we define partial partitioning trees. Roughly speaking, partial partitioning trees of width at most k correspond to tree decompositions such that all *internal* bags are of size at most $k+1$ – the leaves are allowed to have arbitrary size.

Definition 2 (partial partitioning trees). *Given a graph $G = (V, E)$, a* partial partitioning tree *is a couple (T, τ), where T is a tree and τ is a one-to-one function from the set of leaves of T to the parts of some partition $(E_1, \ldots, E_p)$ of E. The bags $\delta(\mu(i))$ of the internal vertices are like in the case of partitioning trees. The bag of the leaf labeled E_i is the set of vertices incident to E_i. The partition $(E_1, \ldots, E_p)$ is called the* displayed partition *of (T, τ).*

The width of a partial partitioning tree is $\max |\delta(\mu(i))|$ *over all* internal *nodes of T.*[4]

Partitioning trees are exactly partial partitioning trees such that the corresponding displayed partition of the edge set is a partition into singletons. Given an arbitrary graph, our aim is to characterize displayed partitions corresponding to partial partitioning trees of width at most k. Actually we only consider *connected* partial partitioning trees, and also the more particular case when the labels of the internal nodes are potential maximal cliques, and we will see that this classes contain the optimal decompositions.

The *connected partial partitioning trees* (strongly related to a similar notion on tree decompositions) are defined as follows. Let X be a set of vertices of G. We say that the set of edges F is a *flap* for X if F is formed by a unique edge with both endpoints in X or if there is a connected component of $G - X$, induced by a vertex subset C of G, and F corresponds exactly to the set of edges of G incident to C (so the edges of F are either between vertices of C or between C and its neighborhood $N_G(C)$). Given a partial partitioning tree (T, τ) and an internal node r which will be considered as the root, let $T(i)$ denote the subtree of T rooted in node i. We denote by $E(i)$ the union of all edge subsets mapped on the leaves of $T(i)$. We say that (T, τ) is a connected partial partitioning tree if and only if, for each internal node j and any son i of j, the edge subset $E(i)$ forms a flap for $\delta(\mu(j))$.

Due to space restrictions, all proofs of this section are given in Appendix A.

Lemma 1. *Given an arbitrary partial partitioning tree (T, τ) of G, there always exists a connected partial partitioning tree (T', τ') such that each edge subset mapped on a leaf of T' is contained in an edge subset mapped on a leaf of T and each bag of (T', τ') is contained in some bag of (T, τ).*

Let $\mathcal{P}$ be a set of partitions of E. We define a new, larger set of partitions $\mathcal{P}^{\uparrow}$ as follows. Initially, $\mathcal{P}^{\uparrow} = \mathcal{P}$. Then, for any partition $\mu = (E_1, E_2, \ldots, E_p) \in \mathcal{P}^{\uparrow}$ and any partition

[4] We emphasys again that a partial partitioning tree might be of small width even if its leaves are mapped on big edge sets.

$\nu = (F_1, F_2, \ldots, F_q, F_{q+1}, \ldots F_r) \in \mathcal{P}$ such that $F_{q+1} \cup \ldots \cup F_r = E_1$, we add to $\mathcal{P}^{\uparrow}$ the partition $\mu \oplus \nu = (E_2, \ldots, E_p, F_{q+1}, \ldots, F_r)$. The process is iterated until it converges. In terms of partial partitioning trees, each partition $\nu \in \mathcal{P}$ is the displayed partition of a partial partitioning tree with a unique internal node. Initially elements of $\mathcal{P}^{\uparrow}$ correspond to these star-like partial partitioning trees. Then, given any partial partitioning tree T_μ corresponding to an element $\mu = (E_1, E_2, \ldots, E_p) \in \mathcal{P}^{\uparrow}$ and a star-like partial partitioning tree T_ν corresponding to an element $\nu = (F_1, F_2, \ldots, F_q, F_{q+1}, \ldots F_r) \in \mathcal{P}$, if the leaf E_1 of the first tree is exactly $F_{q+1} \cup \ldots \cup F_r$, then we glue the two trees by identifying the leaf E_1 of T_μ with the internal node of T_ν and then removing the leaves $F_1, \ldots, F_q$ of T_ν. Thus $\mu \oplus \nu$ is the displayed partition of the new tree. See Figure 1 for an example of this recursive gluing on partial partitioning trees. (A similar but simpler grammar is used in [16].)

The proof of the following statement is an easy consequence of the definitions:

Lemma 2. *Let T be a partial partitioning tree obtained by recursive gluing. The root of T corresponds to the internal node of the first star-like partial partitioning tree used in the recursive gluing. Let x be an internal node of T, and denote by μ_x the partition of $\mathcal{P}$ corresponding to the star-like tree with internal node x – which might be different from the partition $\mu(x)$ introduced in Definition 1. Then for any son y of x, $E(y)$ is a part of μ_x. (Recall that $E(y)$ denotes the union of parts of E mapped on leaves of the subtree $T(y)$ of T rooted in y.) If x is the root, then the sets $E(y)$ for all sons y of x are exactly the parts of μ_x.*

Conversely, let T be a (rooted) partial partitioning tree such that, for each internal node x there is a partition $\mu_x \in \mathcal{P}$ such that for any son y of x, $E(y)$ is a part of μ_x, and, moreover, if x is the root then its sons correspond exactly to the parts of μ_x. Then T is obtained by gluing the partitions μ_x, starting from the root and in a breadth-first search order.

Let $\mathcal{P}_{k-flap}$ be the set of partitions μ of E such that $\delta(\mu)$ is of size at most $k+1$ and the elements of μ are exactly the flaps of $\delta(\mu)$. The set $\mathcal{P}_{k-pmc}$ is the subset of $\mathcal{P}_{k-flap}$ such that for any $\mu \in \mathcal{P}_{k-pmc}$, its border $\delta(\mu)$ is a potential maximal clique. Thus the sets $\mathcal{P}^{\uparrow}_{k-flap}$ and $\mathcal{P}^{\uparrow}_{k-pmc}$ correspond to partial partitioning trees of width at most k.

By Lemma 2, for any graph G, $\mathcal{P}^{\uparrow}_{k-flap}$ is the set of displayed partitions of connected partial partitioning trees of width at most k. Moreover, $\mathcal{P}^{\uparrow}_{k-pmc}$ is the set of displayed partitions of connected partitioning trees of width at most k and such that the bags of all internal vertices are potential maximal cliques. Consequently, we have:

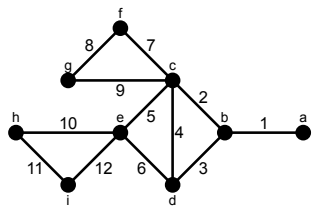
Lemma 3. *G is of treewidth at most k if and only if $\mathcal{P}^{\uparrow}_{k-flap}$ contains the partition into singletons, and if and only if $\mathcal{P}^{\uparrow}_{k-pmc}$ contains the partition into singletons.*

Clearly $\mathcal{P}_{k-flap}$ is of size $O(n^{k+1})$ and $\mathcal{P}_{k-pmc}$ is of size at most the number of potential maximal cliques of the graph.

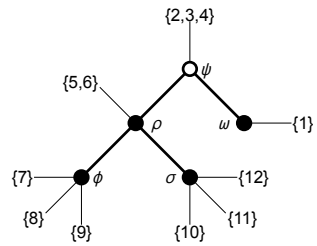
Lemma 4. *Given a set of partitions $\mathcal{P}$ and the corresponding set $\mathcal{P}^{\uparrow}$, we say that $\mathcal{P}^{\uparrow}$ is* orientable *if, for any partition $\mu \in \mathcal{P}^{\uparrow}$ and any part F of μ, there is a partition $\nu \in \mathcal{P}^{\uparrow}$ finer than μ (i.e. each part of ν is contained in a part of μ) and a partial partitioning tree T_ν displaying ν in which the leaf mapped on F is adjacent to the root.*

The sets of partitions $\mathcal{P}^{\uparrow}_{k-flap}$ and $\mathcal{P}^{\uparrow}_{k-pmc}$ are orientable.

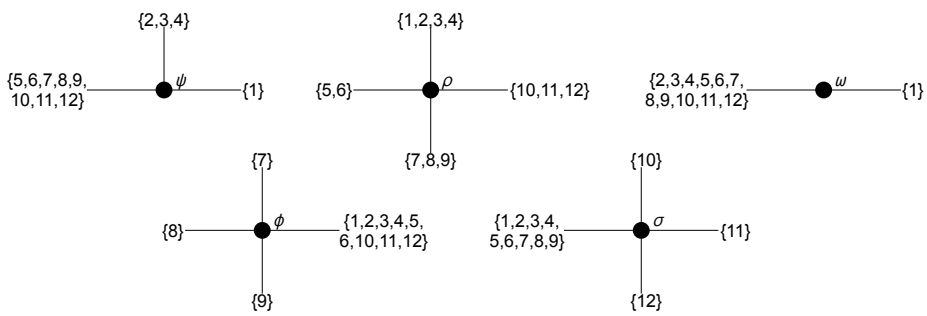
Following [16], we define $\mathcal{P}$-brambles, associated to any set $\mathcal{P}$ of partitions of E.



(a) A graph G = (V,E).



(b) A partial partitioning tree in $\mathcal{P}^{\uparrow}$, with ψ as root. Here, the bag associated to node ψ (in the corresponding classical tree decomposition) would be $\{b, c, d\}$.



(c) The corresponding partial partitioning trees in $\mathcal{P}$.

Fig. 1. The partial partitioning tree in $\mathcal{P}^{\uparrow}$ can be obtained by gluing the given partial partitioning trees in $\mathcal{P}$ in this order: $\psi \oplus \rho \oplus \omega \oplus \phi \oplus \sigma$ (see Lemma 2 and definition of $\oplus$ operator).

Definition 3 (bramble). *Let $\mathcal{P}$ be an arbitrary set of partitions of E. A $\mathcal{P}$-bramble is a set $\mathcal{B}$ of pairwise intersecting subsets of E, all of them of size at least 2, and such that for any partition $\mu = (E_1, \ldots, E_p) \in \mathcal{P}$, there is a part $E_i \in \mathcal{B}$.*

With this definition, one can see that a bramble of order $k+2$ corresponds exactly to a $\mathcal{P}_{k-flap}$-bramble, and a compact bramble of order $k+2$ corresponds to a $\mathcal{P}_{k-pmc}$-bramble.

We say that a set of partitions $\mathcal{P}^{\uparrow}$ is *refining* if for any partitions $(A, A_2, \ldots, A_p)$ and $(B, B_2, \ldots, B_q)$ in $\mathcal{P}^{\uparrow}$, with A and B disjoints, there exists a partition $(C_1, \ldots, C_r)$ in $\mathcal{P}^{\uparrow}$ such that each part C_i is contained in some $A_j, 2 \leq j \leq p$, or in some $B_l, 2 \leq l \leq q$.

In [16], the authors show that for the set $\mathcal{P}_k$ defined by the partitions of E having borders of size at most $k+1$ (without any other restriction), $\mathcal{P}_k^{\uparrow}$ is refining. Using Lemma 1, we can easily deduce that $\mathcal{P}_{k-flap}^{\uparrow}$ is refining. On the other hand, much more efforts are required to prove that $\mathcal{P}_{k-pmc}^{\uparrow}$ is also refining (see Appendix A).

Theorem 3. *For any graph G, the sets of partitions $\mathcal{P}_{k-flap}^{\uparrow}$ and $\mathcal{P}_{k-pmc}^{\uparrow}$ are refining.*

The following result implies the "hard part" of Theorem 1. Indeed, by applying Theorem 4 to $\mathcal{P}_{k-flap}^{\uparrow}$, we have that any graph of treewidth greater than k has a bramble of order $k+2$. For the sake of completion and for better understanding, we give the proof of [16] in Appendix A.

Theorem 4 ([16]). *Let $\mathcal{P}$ be a set of partitions of E and suppose that $\mathcal{P}$ is refining and does not contain the partition into singletons. Let $\mathcal{B}$ be a set of subsets of E such that:*

1. *Each element of $\mathcal{B}$ is of size at least 2 and it is a part of some $\mu \in \mathcal{P}$;*
2. *For each $\mu = (E_1, \ldots, E_p) \in \mathcal{P}$, there is some part $E_i \in \mathcal{B}$;*
3. *$\mathcal{B}$ is upper closed, i.e. for any $F \in \mathcal{B}$, and any superset F' with $F \subset F' \subseteq E$ such that F' is the part of some $\mu' \in \mathcal{P}$, we also have $F' \in \mathcal{B}$;*
4. *$\mathcal{B}$ is inclusion-minimal among all sets satisfying the above conditions.*

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

3 The algorithm

Our goal is to apply Theorem 4 in order to obtain a $\mathcal{P}^{\uparrow}_{k-flap}$-bramble and a $\mathcal{P}^{\uparrow}_{k-pmc}$-bramble. Note that the sets $\mathcal{P}_{k-flap}$ and $\mathcal{P}_{k-pmc}$ are not refining, so we cannot use them directly.

We make an abuse of notation and say that a *flap* of $\mathcal{P}^{\uparrow}$ is a subset of E appearing as the part of some $\mu \in \mathcal{P}^{\uparrow}$. Consequently the flaps of $\mathcal{P}^{\uparrow}$ are exactly the flaps of $\mathcal{P}$. Thus, once we have computed a $\mathcal{P}^{\uparrow}_{k-flap}$-bramble (resp. a $\mathcal{P}^{\uparrow}_{k-pmc}$-bramble), by restricting it to $\mathcal{P}_{k-flap}$ (resp. $\mathcal{P}_{k-pmc}$) we obtain a bramble (resp. a compact bramble) of order $k+2$. The difficulty is that the complexity of our algorithm should be polynomial in the size of $\mathcal{P}_{k-flap}$ (resp. $\mathcal{P}_{k-pmc}$), while the sets $\mathcal{P}^{\uparrow}_{k-flap}$ and $\mathcal{P}^{\uparrow}_{k-pmc}$ may be of exponential size even for small k.

We give now our main algorithmic result. It is stated in a general form, for an arbitrary set of partitions $\mathcal{P}$ such that $\mathcal{P}^{\uparrow}$ is refining and orientable.

Theorem 5 (main theorem). *Let $\mathcal{P}$ be a set of partitions of E. Suppose that $\mathcal{P}^{\uparrow}$ is refining, orientable and does not contain the partition into singletons. Then there is an algorithm constructing a $\mathcal{P}^{\uparrow}$-bramble (and in particular a $\mathcal{P}$-bramble), whose running time is polynomial in the size of E and of $\mathcal{P}$.*

The following algorithm is a straightforward translation of Theorem 4 applied to $\mathcal{P}^{\uparrow}$, so the output $\mathcal{B}_f$ is indeed a $\mathcal{P}^{\uparrow}$-bramble.

```
BRAMBLE(P)
begin
    B ← the set of the flaps of P;
    B_f ← ∅;
    foreach F ∈ B of size one do
        Remove F from B;
    end foreach
    foreach F ∈ B taken in inclusion order do
        if there is a partition μ ∈ P↑ such that F is the unique non-removed flap of the
        partition or ∃F' ∈ B_f : F' ⊆ F then
            Add F to B_f;
        else
            Remove F from B;
        end if
    end foreach
    return B_f;
end
```

Unfortunately the size of $\mathcal{P}^{\uparrow}$ may be exponential in the size of $\mathcal{P}$ and E, and hence the algorithm does not satisfy our complexity requirements because of the test "**if** there is a partition $\mu \in \mathcal{P}^{\uparrow}$ such that F is the unique non-removed flap of the partition", which works on $\mathcal{P}^{\uparrow}$. We would like it to work on $\mathcal{P}$ instead. Thus we replace this test by a marking process working on $\mathcal{P}$ instead of $\mathcal{P}^{\uparrow}$ but giving the same bramble (recall that the flaps of $\mathcal{P}^{\uparrow}$ are exactly the flaps of $\mathcal{P}$). Let us introduce some definitions for the marking. A flap F is said to be *removed* if it has already been removed from the final $\mathcal{P}$-bramble (instruction "Remove F from $\mathcal{B}$"). Intuitively these *removed* flaps induce some forcing among other flaps: some of the flaps must be added to the final bramble, some others cannot be added to the final bramble. Thus whenever a flap is *removed*, we call the algorithm UpdateMarks. We use two types of markings on the flaps: *forbidden* and *forced*. We prove that a flap F will be marked as *forced* if and only if there is some partition $\mu \in \mathcal{P}^{\uparrow}$ such that all flaps of μ, except F, are *removed*. Thus, in Algorithm Bramble, it suffices to test the mark of the flaps.

UpdateMarks
begin
  // marking *forbidden* flaps;
  while $\exists$ *a flap* F *and a partition* $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$ **such that** $(\cup_i^p F_i) = F$ **and** $\forall\ i, 1 \leq i \leq p$, F_i *is* removed **or** F_i *is* forbidden **do**
    Mark F as *forbidden* (if not already marked);
  end while
  // marking *forced* flaps;
  while $\exists\ (F, F_2, \ldots, F_p) \in \mathcal{P}$ **such that** $\forall\ i, 2 \leq i \leq p\ :\ F_i$ *is* removed **or** F_i *is* forbidden **do**
    Mark F as *forced* (if not already marked);
  end while
end

All throughout the algorithm we have the following invariants.

Lemma 5. *A flap F is marked as* forbidden *if and only if there exists a subtree $T(x)$ of a partial partitioning tree T displaying some partition in $\mathcal{P}^{\uparrow}$ such that:*

- *Each flap mapped on a leaf of $T(x)$ is* removed*;*
- *The union of these flaps is exactly F.*

Proof. Suppose first that such a tree exists. We show that the flap F corresponding to the edges mapped on the leaves of $T(x)$ is marked as *forbidden*. Each internal node y of $T(x)$ corresponds to a partition μ_y of $\mathcal{P}$. The edges $E(y)$ of G mapped on the leaves of $T(y)$ form a flap, by construction of T (see also Lemma 2). Consider these internal nodes of $T(x)$, in a bottom up order, we prove by induction that all flaps of such type are marked as *forbidden*. By induction, when node y is considered, for each of its sons $y_1, \ldots, y_p$, either the son is a leaf and hence the corresponding flap is *removed*, or $E(y_i)$ has been previously marked as *forbidden*. Then Algorithm UpdateMarks forbids the flap $E(y)$. Consequently, $E(x)$ is marked as *forbidden*.

Conversely, let F be any flap marked as *forbidden*, we construct a partial partitioning tree T with a node x as required. We proceed by induction on the inclusion order over the

forbidden flaps. When a flap F becomes *forbidden*, it is the union of some flaps $F_1, \ldots, F_p$, each of them being *forbidden* or *removed*, and such that $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q)$ is an element of $\mathcal{P}$. By induction hypothesis, to each F_i, $i \leq p$, we can associate a tree T_i (which is a subpart of a partial partitioning tree) such that the flaps mapped on the leaves of T_i form a partition of F_i and they are all *removed*. Notice that, if F_i is a *removed* flap, then the tree T_i is only a leaf. Consider now a tree $T(x)$ formed by a root x, corresponding to the partition $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$, and linked to the roots of $T_1, \ldots, T_p$. Consider any partition $(F, F'_1, \ldots, F'_r) \in \mathcal{P}$ (such a partition exists, since F is a flap). Note that $\cup_j F'_j = \cup_{i \geq p+1} F_i$. The final tree T is obtained by choosing a root z, corresponding to $(F, F'_1, \ldots, F'_r)$, to which root we glue the subtree T_x by adding the edge xz, and for each flap F'_j we add a leaf adjacent to the root z mapped on F'_j. Thus the final tree T is a gluing between the tree rooted on x and the tree rooted on z. By construction, for each internal node y of this tree, the sets $E(y')$ for the sons y' of y are parts of some partition $\mu_y \in \mathcal{P}$. By Lemma 2, the tree T is a partial partitioning tree displaying an element of $\mathcal{P}^{\uparrow}$. Clearly all leaves of $T(x)$ are mapped on *removed* flaps. □

Lemma 6. *A flap F is marked as* forced *if and only if there is a partition in $\mathcal{P}^{\uparrow}$ such that F is the only non-removed flap of the partition.*

Proof. Let us show that F is the unique non-removed flap of some partition in $\mathcal{P}^{\uparrow}$ if and only if there is a partial partitioning tree T displaying a (possibly another) partition in $\mathcal{P}^{\uparrow}$ such that all leaves but F correspond to *removed* flaps and, moreover, the leaf mapped on F is adjacent to the root of T. Clearly if such a tree exists, the partition μ displayed by the partial partitioning tree has F as the unique non-removed flap. Suppose now that F is a unique non-removed flap of some partition $\mu' \in \mathcal{P}^{\uparrow}$. By the fact that $\mathcal{P}^{\uparrow}$ is orientable, there is a partial partitioning tree T displaying some partition μ, finer than μ', and such that the leaf of T mapped on F is adjacent to the root of the tree. Thus every flap F_i of μ other than F is contained in some flap F'_j of μ'. Since flap F'_j has been *removed* by algorithm Bramble, flap F_i has also been *removed* (in a previous step, unless the trivial case $F_i = F'_j$): indeed flap F_i has been treated by the algorithm before F'_j, and if F_i is not *removed* it means that it has been added to the bramble $\mathcal{B}_f$, and hence the algorithm will not remove any superset of F_i – contradicting the fact that F'_j is now *removed*. We conclude that all flaps of μ, except F, have been *removed*.

It remains to prove that a flap F is marked as *forced* if and only if there is a partial partitioning tree T displaying a partition in $\mathcal{P}^{\uparrow}$, such that all leaves but F correspond to *removed* flaps and, moreover, the leaf mapped on F is adjacent to the root of T.

First, if such a tree exists, let z be its root. Then z corresponds to a partition $(F, F_2, \ldots, F_p)$ in $\mathcal{P}$, and each flap F_i corresponds to a subtree T_i of $T - z$. By Lemma 5, every flap $F_i, 2 \leq i \leq p$, is *removed* or *forbidden*. Then Algorithm UpdateMarks marks the flap F as forced.

Conversely, suppose that Algorithm UpdateMarks marks the flap F as forced. Let $(F, F_2, \ldots, F_p)$ be the partition in $\mathcal{P}$ which has triggered this mark, so all flaps F_i are *removed* or *forbidden*. Thus to each such flap corresponds a subtree T_i of some partial partitioning tree, such that the leaves of T_i form a partition of F_i and they are all *removed* flaps. The tree T, formed by a root x linked to the roots of each T_i, plus a leaf (mapped on F) adjacent to z satisfies our claim. □

Let us discuss now the time complexity of our algorithm. This complexity is the maximum between:

1. the overall complexity of the calls of UpdateMarks;
2. the complexity of the tests "$\exists F' \in \mathcal{B}_f : \ F' \subseteq F$" of the Bramble algorithm.

Clearly both parts are polynomial in the total number of flaps, the number of elements of $\mathcal{P}$ and the size of the graph. The number of flaps is itself at most $m|\mathcal{P}|$, since each partition has at most m parts. It is easy to see that the overall complexity is quadratic in the size of $\mathcal{P}$, times a small polynomial in the size of the graph. This achieves the proof of Theorem 5.

Nevertheless let us go into a little more details, which will allow us to prove that, when we apply our algorithm to $\mathcal{P}_{k-flap}$ and $\mathcal{P}_{k-pmc}$, we obtain better complexity, as claimed in Theorem 2. In particular the running time will be linear in the size of $\mathcal{P}_{k-flap}$ (resp. $\mathcal{P}_{k-pmc}$), times a small polynomial in n. The overall complexity of UpdateMarks is given by the complexity of updating the *forbidden* flaps. Let us say that a couple (F, μ), formed by a flap F and a partition $\mu \in \mathcal{P}$ is a *good couple* if μ is of the form $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q)$ with $F = F_1 \cup \ldots \cup F_p$. We use the following data structure. Each flap F_i points towards each good couple (F_i, μ_i). To each good couple (F_i, μ_i) we associate the list of all flaps F'_j which are parts of μ_i and contained in F_i, and we call it the list of *good subflaps*; this list is of size at most m. Whenever a flap F'_j is triggered as *forbidden* or *removed*, it warns all good couples (F_i, μ_i) to which it is associated (in the list of good subflaps). When, for a couple (F_i, μ_i), all good subflaps of the associated list have become *removed* or *forbidden*, the flap F_i is also marked as *forbidden* and the process continues. By Lemma 5 and by induction on the inclusion order of the flaps, the algorithm correctly marks the *forbidden* flaps as required. The complexity of the whole marking process of *forbidden* flaps is $O(m \cdot \texttt{number of good couples})$. Thus the overall complexity of UpdateMarks is also $O(m \cdot \texttt{number of good couples})$ plus the complexity of computing our data structure of good couples and good associated subflaps, which is also the number of good couples, multiplied by a small polynomial in the size of the graph. We discuss below this complexity for each particular case.

We also have to test, in Algorithm Bramble, whether a flap F contains some flap $F' \in \mathcal{B}_f$. For this purpose we construct a directed acyclic graph G_{flaps} such that each node of this graph is a (non-trivial) flap of $\mathcal{P}$ and the transitive closure of the graph is the inclusion relation between flaps. Whenever a flap F' is put into $\mathcal{B}_f$, it marks all flaps F such that there exists a path from F' to F in G_{flaps} as *forced by inclusion*. With standard techniques, this marking is linear in the size of G_{flaps}, so it remains to ensure that the number of arcs of G_{flaps} is moderate, which we also prove on each particular case (for further details, see Appendix B).

Theorem 5 has been stated in its most general form, and as we shall discuss in the conclusion it can be used for other parameters than treewidth. In the case of treewidth, algorithm Bramble and the UpdateMarks procedure can be further refined in order to obtain brambles and compact brambles of order $k+2$, for graphs of treewidth larger than k. The running time will be, as announced in Theorem 2, $O(n^{k+4})$ for brambles and $O(n^4 r^2)$ for compact brambles, where r is the number of minimal separators of the input graph.

We point out that Algorithm UpdateMarks can be seen as a gereralization of the algorithms of [2, 10, 14]. Indeed the algorithm of Arnborg et al. [2], deciding whether a graph has treewidth at most k in $O(n^{k+2})$, behaves like when we call UpdateMarks on the set $\mathcal{P}_{k-flap}$, only once, right after removing all trivial flaps (formed by a unique edge). Thus a graph is of treewidth at most k if and only if, after this run, the flap formed by the whole edge set (which is the unique flap of the trivial partition) is marked as *forbidden*. In a similar way, if we apply algorithm UpdateMarks on $\mathcal{P}_{k-pmc}$ (to which we must add the trivial partition of E), it behaves like the algorithms of [10, 14]. Therefore it is not surprising that the data

structures used in [2, 14] can also be used in our case in order to fasten the marking algorithm. In both cases the key idea is that the number of good couples (or at least good couples that we really need to use) is moderate. The number of good couples is $O(n^{k+3})$ for $\mathcal{P}^{\uparrow}_{k-flap}$ and $O(n \cdot \texttt{number of potential maximal cliques})$ for $\mathcal{P}^{\uparrow}_{k-pmc}$. Moreover, the graph G_{flaps} can be computed in $O(n^{k+4})$ for $\mathcal{P}^{\uparrow}_{k-flap}$, and in $O(n^4 r^2)$ for $\mathcal{P}^{\uparrow}_{k-pmc}$. Due to space restrictions, the proofs are given in Appendix B.

4 Conclusion

We have presented in this article an algorithm computing brambles of large order for arbitrary graphs. The running time for the algorithm is $O(n^{k+4})$ for computing a bramble of order $k+2$, and of course we cannot expect drastic improvements since the size of the bramble itself is of order $\Omega(n^{k+1})$. There are other equivalent definitions for brambles. One of the most popular ones consists in defining a bramble of G as a set of pairwise touching connected sugraphs of G (we recall that two subgraphs touch if they share a vertex or if there is an edge between the two). The order of the bramble is the minimum size of a hitting set, i.e. of a vertex subset intersecting each of these connected subgraphs. It is clear that the two definitions are equivalent (see also [6]). The latter definition allows, but only in some particular cases, to consider brambles of smaller size. For planar $p \times p$ grids, the "crosses" (a line plus column) form a bramble of order p; see also [7] for other constructions. Treewidth can be defined in terms of graph searching as a game between cops and a robber. As in many games, we can consider the graph of all possible configurations (here it has $\Theta(n^{k+2})$ vertices) and it is possible to compute [13] which are winning configurations for the cops (treewidth at most k) and which are winning for the robber (treewidth larger than k). This can also be considered as a certificate for large treewidth, but clearly more complicated than brambles. Eventually, one can also argue that graphs of treewidth at most k can be characterized by $f(k)$ obstructions for some function f – so the size does not depend on k. This is a consequence of the Robertson and Seymour's famous graph minor theorem, and also directly from the fact that the treewidth problem is fixed parameter tractable. Nevertheless, this function $f(k)$ can be extremely huge. To our knowledge, the problem of defining good obstructions to tree decompositions – in the sense that these obstructions should be of moderate size and easy to manipulate – is largely open.

Another interesting question is whether our algorithm for computing brambles can be used for other tree-like decompositions. As we noted, other parameters (branchwidth, pathwidth, rankwidth...) fit into the framework of [1, 16] of partitioning trees and we can define $\mathcal{P}^{\uparrow}_{k-xxx-width}$-brambles in similar ways. The problem is that the size of "basic partitions" (the equivalent of our set $\mathcal{P}_{k-flap}$) may be exponential in n even for small k. Due to results of [17, 18], for branchwidth we can also restrict to connected decompositions and thus our algorithm can be used in this case with similar complexity as for treewidth.

Acknowledgement. I.Todinca wishes to thank Fedor Fomin for fruitful discussions on this subject.

References

1. O. Amini, F. Mazoit, S. Thomassé, and N. Nisse. Partition submodular functions, 2008. http://www.lirmm.fr/ thomasse/liste/partsub.pdf.
2. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. on Algebraic and Discrete Methods*, 8:277–284, 1987.
3. E. H. Bachoore and H. L. Bodlaender. New upper bound heuristics for treewidth. In *Experimental and Efficient Algorithms, 4th InternationalWorkshop, WEA 2005*, volume 3503 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2005.
4. P. Bellenbaum and R. Diestel. Two short proofs concerning tree-decompositions. *Combinatorics, Probability & Computing*, 11(6), 2002.
5. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Computing*, 25:1305–1317, 1996.
6. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.
7. H. L. Bodlaender, A. Grigoriev, and A. M. C. A. Koster. Treewidth lower bounds with brambles. *Algorithmica*, 51(1):81–98, 2008.
8. H. L. Bodlaender and A. M. C. A. Koster. On the maximum cardinality search lower bound for treewidth. *Discrete Applied Mathematics*, 155(11):1348–1372, 2007.
9. H. L. Bodlaender, T. Wolle, and A. M. C. A. Koster. Contraction and treewidth lower bounds. *J. Graph Algorithms Appl.*, 10(1):5–49, 2006.
10. V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: grouping the minimal separators. *SIAM J. on Computing*, 31(1):212 – 232, 2001.
11. V. Bouchitté and I. Todinca. Listing all potential maximal cliques of a graph. *Theoretical Computer Science*, 276(1-2):17–32, 2002.
12. F. Clautiaux, J. Carlier, A. Moukrim, and S. Nègre. New lower and upper bounds for graph treewidth. In *Experimental and Efficient Algorithms, Second International Workshop, WEA 2003*, volume 2647 of *Lecture Notes in Computer Science*, pages 70–80. Springer, 2003.
13. F. V. Fomin, P. Fraigniaud, and N. Nisse. Nondeterministic graph searching: From pathwidth to treewidth. *Algorithmica*, 53(3):358–373, 2009.
14. F. V. Fomin, D. Kratsch, I. Todinca, and Y. Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM J. Comput.*, 38(3):1058–1079, 2008.
15. F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.
16. L. Lyaudet, F. Mazoit, and S. Thomassé. Partitions versus sets: A case of duality. Submitted, 2009. http://www.lirmm.fr/ thomasse/liste/dualite.pdf.
17. F. Mazoit. *Décompositions algorithmiques des graphes.* PhD thesis, Ecole normale supérieure de Lyon, 2004. In French.
18. F. Mazoit. The branch-width of circular-arc graphs. In *Proceedings of the 7th Latin American Theoretical Informatics Symposium (LATIN 2006)*, volume 3887 of *Lecture Notes in Computer Science*, pages 727–736. Springer-Verlag, 2006.
19. N. Robertson and P. D. Seymour. Graph minors X. Obstructions to tree decompositions. *Journal of Combinatorial Theory Series B*, 52:153–190, 1991.
20. P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *J. Comb. Theory, Ser. B*, 58(1):22–33, 1993.

A Proofs for Section 2

Proof of Lemma 1. Let (T, τ) be an arbitrary partial partitioning tree of G. Let us construct the desired connected partial partitioning tree (T', τ').

Take an internal node i of T other than the root, j its father, and consider the edge subset $E(i)$ (the union of all edge subsets mapped on the leaves of $T(i)$). If this set forms a flap for $\delta_T(\mu(j))$, there is nothing to do. If not, then it forms several flaps $C_1, \ldots, C_p$, each of these flaps C_l containing one ore more leaves $e_l^1, \ldots, e_l^q$ of $T(i)$. For each flap C_l, take the subtree $T(i)(l)$ of $T(i)$ with only the leaves e_l, and re-root it to i. In this new tree T', i has as many sons as the number of flaps C_l formed by the edge subset $E(i)$ in T. The set $\delta_{T'}(\mu(j))$ in T' is the same as the set $\delta_T(\mu(j))$ in T. Indeed, the set of edges mapped on the leaves of linked subtrees $T(i)(l)$ are flaps, so they are mutually disjoints, and hence no edge is added to $\delta_{T'}(\mu(j))$. Moreover, for every flap C_l, the set of edges shared with some other flaps in some $E'(i)$ (other than $E(i)$) does not change. So the width of the partial partitioning tree (T', τ') being constructed does not increase. By doing this for every internal node of T, we obtain a connected partial partitioning tree (T', τ') such that each bag is contained in some bag of (T, τ), and such that (T, τ) and (T', τ') have the same width. □

We point out that the connected partial partitioning trees are strongly related to *connected tree decompositions*:

Definition 4. *We say that a tree decomposition TD is connected if it can be rooted such that, for each node y other than the root, the set $C(y)$ of vertices appearing in bags of the subtree $TD(y)$, but not appearing in the bag $X(x)$ of the father x of y, induces a connected subgraph in G.*

Actually $C(y)$ is one of the components of $G - X(x)$. Any tree decomposition can be transformed into a connected one, such that the bags of the latter are contained in the bags of the first [2]. Moreover the leaf bags of the connected tree decomposition are contained into leaf bags of the initial one.

Some properties of potential maximal cliques. Tree decompositions of a graph G are often expressed in terms of *triangulations*. A graph H with the same vertex set as G is called a *triangulation* of G if the edge set of G is contained in the edge set of H and H is chordal (i.e. each cycle of length at least four of H has an edge between non-consecutive vertices). Every chordal graph has a tree decomposition such that the bags of the decomposition correspond to the maximal cliques of H. Conversely, consider a tree decomposition TD of G such that no bag is strictly contained into another. If we complete each bag into a clique (i.e. we add, in the graph G, all missing edges inside the bag), we obtain a triangulation H_{TD} of G and the maximal cliques of H are exactly the bags of G. Therefore the treewidth of G can be defined as the minimum clique-size of H minus one, over all triangulations H of G. Clearly, we can restrict to *minimal* triangulations. A triangulation H of G is said to be minimal if no strict subgraph of H is a triangulation of G.

A set of vertices K of G is called a *potential maximal clique* [10] if there is a minimal triangulation H of G such that K induces a maximal clique in H. In other terms, the potential maximal cliques correspond to the bags of minimal triangulations.

Let $G[K]$ denote the subgraph of G induced by K. Denote by $\mathcal{C}(K) = (C_1, \ldots, C_p)$ the connected components of $G - K$ and let $S_i = N(C_i)$ be the neighborhood of C_i in G. The following statement is a simple characterization of potential maximal cliques.

Theorem 6 ([10]). *Let K be a set of vertices of G. Then K is a potential maximal clique if and only if:*

- *For any $C_i \in \mathcal{C}(K)$, its neighborhood S_i is strictly contained in K;*
- *The graph $G[K]_{\{S_1,\ldots,S_p\}}$, obtained from $G[K]$ by completing each S_i into a clique, is a complete graph.*

Potential maximal cliques are strongly related to *minimal separators*. A set of vetices S of G is a *minimal separator* if there are at least two distinct connected components C and D of $G - S$ such that $S = N(C) = N(D)$. Indeed, if K is a potential maximal clique, then the sets S_i as above are exactly the minimal separators of G, contained in K [10].

Moreover the number of potential maximal cliques of a graph is related to the number of minimal separators.

Theorem 7 ([11]). *Let G be a graph with n vertices, π be the number of potential maximal cliques of G and r be the number of its minimal separators. Then $\pi \leq nr^2 + nr + 1$ and, moreover, all potential maximal cliques can be computed in $O(n^4r^2)$ time.*

Several classes of graphs are known for having a "small" number of minimal separators and potential maximal cliques, in the sense that these quantities are polynomial in the size of the graph. For example, permutation graphs have $O(n)$ minimal separators and potential maximal cliques, circular and circular-arc and weakly chordal graphs have $O(n^2)$ minimal separators and $O(n^3)$ potential maximal cliques – see [10] for a survey.

We also know [14] that, given a graph, the treewidth can be computed in $O(n^3\pi)$ time, or in $O(n^4r^2)$ time.

Discussion on Lemma 4. To prove that $\mathcal{P}^{\uparrow}_{k-flap}$ is orientable, we use the same ideas as in the proof of Lemma 1. We want to prove that for any partition $\mu \in \mathcal{P}^{\uparrow}_{k-flap}$ and any part F of μ, there is a partition $\nu \in \mathcal{P}^{\uparrow}_{k-flap}$ finer than μ and a partial partitioning tree T_ν displaying ν in which the leaf mapped on F is adjacent to the root.

Note that the partial partitioning tree of any partition in $\mathcal{P}^{\uparrow}_{k-flap}$ is connected, since every part of any partition in $\mathcal{P}_{k-flap}$ is a flap. Let T_μ be a (connected) partial partitioning tree displaying $\mu \in \mathcal{P}^{\uparrow}_{k-flap}$. The construction in the proof of Lemma 1 can be done for any chosen initial root. Thus one can construct a (connected) partial partitioning tree T_ν displaying a partition ν finer than μ, and such that the root of T_ν is adjacent to the leaf mapped on F. Moreover the bags corresponding to internal nodes of T_ν are exactly the bags corresponding to internal nodes of T_μ, and hence the displayed partition ν of T_ν is exactly the initial partition μ (in $\mathcal{P}^{\uparrow}_{k-flap}$). In particular this proves that $\mathcal{P}^{\uparrow}_{k-pmc}$ is also orientable. □

Proof of Theorem 3. Let $G = (V, E)$ be a graph. Let $\mathcal{P}_k$ be the set of partitions of E having borders of size at most $k + 1$. By [16], $\mathcal{P}^{\uparrow}_k$ is refining.

$\mathcal{P}_{k-flap}$ is a subset of $\mathcal{P}_k$. Indeed, every partition in $\mathcal{P}_{k-flap}$ has borders of size at most $k+1$, with the extra restriction that every part of the partition is a flap. Thus, by the construction of $\mathcal{P}^{\uparrow}$ sets, $\mathcal{P}^{\uparrow}_{k-flap}$ is a subset of $\mathcal{P}^{\uparrow}_k$.

Let $(A, A_1, \ldots, A_p)$ and $(B, B_1, \ldots, B_q)$ be two partitions in $\mathcal{P}^{\uparrow}_{k-flap}$, which are also in $\mathcal{P}^{\uparrow}_k$, such that $A \cap B = \emptyset$. Let (T_A, τ_A) and (T_B, τ_B) be two partial partitioning trees, with displayed partitions $(A, A_1, \ldots, A_p)$ and $(B, B_1, \ldots, B_q)$ respectively. As $\mathcal{P}^{\uparrow}_k$ is refining, there

exists a partition $(C_1, \ldots, C_r) \in \mathcal{P}_k^{\uparrow}$ finer than $(A_1, \ldots, A_p, B_1, \ldots, B_q)$. Consider a partial partitioning tree (T, τ) with $(C_1, \ldots, C_r)$ as its displayed partition. By Lemma 1, there exists a connected partial partitioning tree (T', τ') such that each edge subset mapped on a leaf of T' is contained in an edge subset mapped on a leaf of T and each bag of (T', τ') is contained in some bag of (T, τ). The displayed partition of (T', τ') is in $\mathcal{P}_{k-flap}^{\uparrow}$, and hence $\mathcal{P}_{k-flap}^{\uparrow}$ contains a partition finer than $(A_1, \ldots, A_p, B_1, \ldots, B_q)$. Thus $\mathcal{P}_{k-flap}^{\uparrow}$ is refining.

Unfortunately, for proving that $\mathcal{P}_{k-pmc}^{\uparrow}$ is refining, we don't have the equivalent of Lemma 1 to simply deduce this from previous results. Indeed, there are small examples showing that in Lemma 1, we cannot add the extra condition that any internal bag of (T', τ') is a potential maximal clique.

We sketch here the proof that $\mathcal{P}_{k-pmc}^{\uparrow}$ is refining. Actually we describe with more details one of the techniques proving that $\mathcal{P}_{k-flap}^{\uparrow}$ is refining, then we adapt this proof to the case of potential maximal cliques.

Firstly, let us have a closer look at the relationship between a connected partial partitioning tree T_μ displaying a partition $\mu \in \mathcal{P}_{k-pmc}^{\uparrow}$ and the associated tree decomposition. Consider this tree decomposition associated to T_μ and remove trivial leaf bags (corresponding to a trivial leaf flap). The new tree decomposition TD_μ has the following properties: (1) all its internal bags are potential maximal cliques, and (2) for each leaf bag X, the intersection S between X and its neighbour X' is a minimal separator of G, and $X \setminus X'$ is a connected component of $G - X'$ such that $S = N_G(X \setminus X')$ (this can be easily deduced from Theorem 6). A tree-decomposition of graph G satisfying these two properties will be called *internally-minimal*. Conversely, by transforming an internally-minimal tree decomposition into a partial partitioning tree, we obtain a tree displaying an element of $\mathcal{P}_{k-pmc}^{\uparrow}$.

Note that each leaf bag of the tree decomposition TD_μ is either a potential maximal clique of size at most $k+1$, or the set of vertices incident to a non-trivial flap of μ. Conversely, when we transform an internally-minimal tree decomposition TD into a partition $\mu \in \mathcal{P}_{k-pmc}^{\uparrow}$, each not-trivial flap of μ corresponds to a leaf bag of the tree decomposition (the bag is formed by the vertices of G incident to the flap).

The following statement, which can be found by a little digging in the proofs of [4], can be considered as the equivalent of Theorem 3 to tree decompositions.

Theorem 8 (gluing theorem [4]). *Let TD_1 and TD_2 be two tree decompositions of G. Consider a leaf bag X of TD_1 and a leaf bag Y of TD_2. Suppose that X and Y are non-crossing, in the sense that X (resp. Y) intersects at most one component of $G - Y$ (resp. $G - X$).*

Then there is a tree decomposition TD of G, whose decomposition tree is obtained from the union TD_1 and TD_2 by identifying the two leaves X and Y, and such that any bag Z of TD is at least as small as the corresponding bag of TD_1 or TD_2. Moreover, when the internal bags of TD_1 and TD_2 are of size at most $k+1$, then the internal bags of TD are also of size at most $k+1$.

Proof. Let S be a minimum-size X, Y-separator of G (i.e. $\forall x \in X$, $\forall y \in Y$, x and y are not connected in $G - S$). By Menger's theorem, there is a set P_X (resp. P_Y) of disjoint paths of G, linking each vertex of S to some vertex of X (resp. of Y). Let D_X (resp. D_Y) be the union of connected components of $G - S$ intersecting X (resp. Y). First we construct a tree decomposition TD_1' of $G - D_X$, having the same tree as TD_1, such that each bag of TD_1' is smaller than the corresponding bag of TD_1 and such that the leaf whose bag is X in TD_1

has bag S in TD'_1. Let $X(i)$ be some bag of TD_1, corresponding to some node i. In TD'_1, the corresponding bag $X'(i)$ is obtained from $X(i)$ as follows. First we remove from $X(i)$ all vertices of D_X. Then for any vertex $y \in D_X \cap X(i)$ such that y is on one of the paths of P_X, we replace y by the endpoint of this path in S. It is easy to check that $|X'(i)| \leq |X(i)|$, that bag X is replaced by S and one can check (see [4] for details) that TD'_1 is a tree decomposition of $G - D_X$. We point out that for each bag $X(i)$, the vertices of $X'(i) \setminus X(i)$ are exactly the vertices s of S such that $X(i)$ separates s and X in the graph G.

The same operation is performed in order to transform TD_2 into a tree decomposition TD'_2, where bag Y has been replaced by S. Eventually TD'_1 and TD'_2 are glued by identifying the two bags S to obtain the tree decomposition TD. □

We now apply the gluing theorem to two internally-minimal tree decompositions TD_1 and TD_2. The tree decomposition TD is not necessarily internally-minimal, but as we shall see it can be refined into an internally-minimal one. Our first remark is that the minimum-size X, Y-separator S of G (see the proof of Theorem 8 for the notations) is also a minimal separator. More precisely, D_X (resp. D_Y) is a connected component of $G - S$ and $N_G(D_X) = N_G(D_Y) = S$.

Let i be a leaf of TD_1 (different from the leaf used for the gluing), we investigate the structure of the bag $X_{TD}(i)$ in the tree decomposition TD. Since TD_1 is internally-minimal, the bag $X_{TD_1}(i)$ is the disjoint union of a minimal separator $S(i)$ and a connected component $C(i)$ of $G - S(i)$ such that $S(i) = N_G(C(i))$. Recall that, after gluing, the bag $X_{TD}(i)$ is obtained from the bag $X_{TD_1}(i)$ by removing the vertices of $X_{TD_1}(i)$ which are in D_X and by adding the vertices of S such that $X_{TD_1}(i)$ separates X from these vertices in graph G. Actually in this case there are no added vertices, so we also have that the leaf bags of TD are *contained* in the leaf bags of TD_1 and TD_2 (not only of smaller size). Let $U(i)$ be the set of vertices of $X_{TD}(i) \cap (S \cup S(i))$. These are the only vertices of $X_{TD}(i)$ which can have neighbors outside $X_{TD}(i)$, in the graph G. Thus $X_{TD}(i) \setminus U(i)$ is a union of connected components $D^1(i), \ldots, D^p(i)$ of $G - U(i)$. For any such component $D^j(i)$, its neighborhood $W^j(i) = N_G(D^j(i))$ is a minimal separator of G (see e.g. [11]). We transform the tree decomposition TD in a new one, by giving the bag $U(i)$ to node i and adding, for each component $D^j(i)$, $1 \leq j \leq p$, a new leaf i^j with bag $D^j(i) \cup W^j(i)$. By doing this on all leaves i of the tree decomposition TD we obtain a new tree decomposition TD', finer that TD, such that each leaf bag of TD' is contained in some leaf bag of TD. Moreover, TD' has well-formed leaves, in the following sense: each leaf bag of TD' is a union of a minimal separator W and a connected component D of $G - W$, such that $N_G(D) = W$. We show how to transform such a tree decomposition into an internally-minimal one, by preserving the set of leaves. For each leaf bag $D \cup W$ we delete from G the set D and we complete the minimal separator W into a clique (the sets D are pairwise disjoint). Let G^- be the graph obtained by these operations. Also note that TD', minus its leaves of type $D \cup W$, is a tree decomposition of G^-. Take now a minimal triangulation of G^- such that each bag of the corresponding tree decomposition TD^- is contained in some internal bag of TD'. It is not hard to check, using Theorem 6, that each potential maximal clique of G^- is also a potential maximal clique of G (see also [10]). We construct an internally-minimal tree decomposition TD'' of G as follows. Fix a root of TD^- and transform it into a connected tree decomposition. For any leaf bag $D \cup W$ of TD', take the highest node of TD^- which contains W; such a node exists, since W induces a clique in G^-. Add a leaf to this node, with bag $D \cup W$. Consequently, TD'' is internally-minimal. The leaves of TD'' of type $W \cup D$ are contained in some leaf of TD' (and of TD); there might also

be some other leaves, corresponding to nodes of TD^-, in which case the corresponding bags are potential maximal cliques of G.

We are now ready to prove that $\mathcal{P}^{\uparrow}_{k-pmc}$ is refining. Let $\mu_1 = (A, A_1, \ldots, A_p)$ and $\mu_2 = (B, B_1, \ldots, B_q)$ be two partitions in $\mathcal{P}^{\uparrow}_{k-pmc}$, with $A \cap B = \emptyset$. Let TD_1 and TD_2 be two internally-minimal tree decompositions corresponding to partial partitioning trees of these partitions (after removing trivial bags, unless A and/or B are trivial, in which case we also keep leaf bags corresponding to these trivial flaps). We apply the gluing theorem on the bags corresponding to flaps A and B and, using the previous observations, we eventually obtain an internally-minimal tree decomposition TD. Each leaf bag of TD is either a potential maximal clique of size at most $k+1$ of G, or it is of type $W \cup D$ and moreover the bag is included in a leaf bag of TD_1 or TD_2. In the latter case, the flap F_D formed by the edges of G incident to component D is included in some $A_i, 1 \leq i \leq p$ or some $B_j, 1 \leq j \leq q$. Therefore the partition $\mu \in \mathcal{P}^{\uparrow}_{k-pmc}$ corresponding to the tree decomposition TD is a refining of $(A_1, \ldots, A_p, B_1, \ldots, B_q)$.

This achieves the proof of Theorem 3. □

Proof of Theorem 4. We give here the proof of [16].

Let us note first that $\mathcal{B}$ always exists: indeed take the set $\mathcal{B}_0$ corresponding to all parts of elements of $\mathcal{P}$, except the singletons. It satisfies all conditions of the theorem, except the minimality. Thus it is sufficient to extract an inclusion-minimal $\mathcal{B} \subseteq \mathcal{B}_0$ satisfying conditions 2 and 3. We prove that such inclusion-minimal $\mathcal{B}$ is a $\mathcal{P}$-bramble.

Suppose that $\mathcal{B}$ is not a $\mathcal{P}$-bramble. Let A, B be two minimal disjoint sets in $\mathcal{B}$. Since $\mathcal{B}$ is minimal, there exists two partitions $(A', A_2, \ldots, A_p), (B', B_2, \ldots, B_q) \in \mathcal{P}^{\uparrow}$ with only $A', B' \in \mathcal{B}$ and such that $A' \subseteq A$, $B' \subseteq B$. As $\mathcal{P}^{\uparrow}$ is refining, there exists a partition $(C_1, \ldots, C_r)$ finer than $(A_2, \ldots, A_p, B_2, \ldots, B_q)$. As $\mathcal{B}$ contains a part of every partition in $\mathcal{P}^{\uparrow}$, $\mathcal{B}$ contains some C_i, which is a subset of some A_j (or B_k). Thus, as $\mathcal{B}$ is upper closed, it contains this A_j (or B_k) – a contradiction. □

B The Bramble algorithm: complexity details

Let us detail first the case of "usual" brambles, when we work with $\mathcal{P}_{k-flap}$. Recall that our UpdateMarks algorithm must ensure that, like in Lemma 5, a flap F will be marked as *forbidden* if and only if there is a partial partitioning tree T with a subtree $T(x)$ such that all flaps mapped on leaves of $T(x)$ are *removed* and the union of these flaps is F. We want to restrict to particular types of partitioning trees, which will allow to prove that we can use $O(n^{k+3})$ good couples.

Let us sketch some of the results of Arnborg et al. [2], on tree decompositions. Recall that we can restrict to connected tree decompositions (see Definition 4 in Appendix A). Given a rooted tree decomposition TD and a node x of TD, we denote by $C_{TD}(x)$ the set of vertices of G appearing in the bags of the subtree $TD(x)$, but not in the bag of the father of x. A connected tree decomposition TD is transformed again into a connected tree decomposition TD', such that each bag of TD' is contained in some bag of T', each component $C_{TD}(x)$ is also a component $C_{TD'}(x')$ for some node x' of TD', the bag roots of the two trees are the same and, most important, we have the following technical condition: for each node y of TD' other than the root, its bag $X_{TD'}(y)$ is formed by the neighborhood of $C_{TD'}(y)$ in the graph G, plus one vertex. In terms of partial partitioning trees, if we denote by $E(y)$ the set of edges

having at least one endpoint in $C_{TD'}(y)$, this means that the bag $X_{TD'}(j)$ is the border of the partition $(E(j), E \setminus E(j))$ of E, plus one vertex. This last result can be read as follows:

Lemma 7. *Let T be a connected partitioning tree corresponding to some element of $\mathcal{P}^{\uparrow}_{k-flap}$. We can transform the tree T into another connected partitioning tree T', corresponding to the same element of $\mathcal{P}^{\uparrow}_{k-flap}$ which has the following properties:*

1. *The roots of T' and T have the same associated partition;*
2. *For each node x of T different from the root, there is a node x' of T' such that $E_T(x) = E_{T'}(x')$ and the flaps mapped on the leaves of $T(x)$ are exactly the flaps mapped on the leaves of $T'(x)$ ($E_T(x)$ denotes the union of flaps mapped on leaves of $T(x)$ in the tree T);*
3. *For each internal node y of T', different from the root, we have that $\delta(\mu_{T'}(y))$ is exactly $\delta((E_{T'}(y), E \setminus E_{T'}(y)))$ plus one vertex.*

By Lemma 7, in our algorithm UpdateMarks it is sufficient to consider good couples of type $(E_{T'}(y), \mu_{T'}(y))$ as described in the lemma. Thus we restrict to good couples (F, μ) such that $\delta(\mu)$ corresponds to $\delta(F, E \setminus F)$ plus one vertex; moreover we may assume that flap F is non-trivial (not reduced to a single edge) since trivial flaps are removed at the initialization step. We claim that the number of such flaps is $O(n^{k+3})$. The number of elements of $\mathcal{P}_{k-flap}$ is $O(n^{k+1})$, since we only consider partitions $\mu = (F_1, \ldots, F_p)$ where $|\delta(\mu)| \leq k+1$ and each part is either an edge with boths ends in $\delta(\mu)$, or it is formed by the set of edges of the input graph G, incident to a connected component of $G - \delta(\mu)$. For any partition μ, the number of non-trivial flaps is at most n, thus the number of all useful flaps is $m + O(n^{k+2})$. To each non-trivial flap F we associate at most n good couples of the form $\delta(F, E \setminus F)$ plus one vertex, which achieves the proof on the number of good couples. For each good couple (F, μ) there are at most m good associated subflaps (flaps of μ, contained in F). But here again we only need to memorize the non-trivial ones, which reduces the list to at most n elements. Altogether, the UpdateMarks algorithms works in $O(n^{k+3})$ time, and computing the list of good flaps and the associated good (non-trivial) subflaps can be done in $O(n^{k+4})$ time.

It remains to discuss how we test, in Algorithm Bramble, whether a flap F contains some flap $F' \in \mathcal{B}_f$. Recall that we want to construct a graph G_{flaps} such that the vertex set of G_{flaps} is the set of all flaps of $\mathcal{P}_{k-flap}$ and the arcs are such that the transitive closure of G_{flaps} gives the inclusion order on the set of flaps. Again using the same ideas as [2], we can show that for any two flaps $F' \subset F$, there is a flap F'' such that $F' \subseteq F'' \subset F$ and, moreover, F is a flap associated to a partition μ, where $\delta(\mu)$ is formed by $\delta(F, E \setminus F)$ plus one vertex. Thus the arcs of the graph G_{flaps} will only be put between flaps (F'', F) with this property. Again for a flap F we consider at most n partition μ, each of them giving at most n flaps F', hence flap F has at most n^2 incoming arcs. The size of G_{flaps} is $O(n^{k+3})$ and the time complexity for computing this graph is $O(n^{k+4})$. Altogether, computing a bramble of order $k+2$ costs $O(n^{k+4})$ time.

A very similar argument holds for potential maximal cliques. Recall that any partial partitioning tree of width at most k can be transformed into a connected partial partitioning tree of the same width and such that, for each internal node, the border of the corresponding partition (i.e. the bag of the node) is a potential maximal clique. Thus can restrict to good couples (F, μ) such that $\delta(\mu)$ corresponds to a potential maximal clique.

For a fixed partition $\mu \in \mathcal{P}_{k-pmc}$, T_μ has at most n leaves corresponding to n disjoint flaps (in $G \setminus \delta(\mu)$), and hence the number of good couples (F, μ) for this fixed μ is at most n. The number of all good couples (F, μ) such that $\delta(\mu)$ corresponds to a potential maximal

clique is $n\pi$ (π is the number of potential maximal clique of G). By [14], for a fixed potential maximal clique $\Omega = \delta(\mu)$ (for some $\mu \in \mathcal{P}_{k-pmc}$), there are at most n minimal separators $S \subset \Omega$, and for each of these minimal separators, there is a unique component C of $G \setminus S$ such that $\Omega \subseteq S \cup C$. So, for a good couple (F, μ), the number of good associated subflaps (flaps in μ, contained in F) is at most n. By [11], a graph G with r minimal separators has $O(nr^2)$ potential maximal cliques, and these potential maximal cliques can be enumerated in $O(n^2r^2)$ time. Thus computing all good couples (F, μ) and their associated lists of good subflaps can be done in $O(n^4r^2)$.

It remains to discuss how we test whether a flap F contains some flap $F' \in \mathcal{B}_f$. For this test, we want to compute a graph G_{flaps}, with good couples (F, μ) as vertices, and such that its transitive closure gives the inclusion order on the set of good couples. For two couples (F, μ) and (F', μ'), we have by [14] that $(F', \mu') \prec (F, \mu)$ (where $\prec$ is the inclusion order on the set of good couples) if $F' \cup \delta(\mu') \subset F \cup \delta(\mu)$ and F' is a good subflap for (F, μ). Thus, in the graph G_{flaps}, we only put an arc between these good couples $((F', \mu'), (F, \mu))$. A good couple (F, μ) has at most n good subflaps, and each good subflap F' consider at most n partitions μ'. Hence a good couple (F, μ) has at most n^2 incoming arcs in G_{flaps}. The size of G_{flaps} is $O(n^2r^2)$, and computing this graph can be done in $O(n^4r^2)$. Altogether, computing a compact bramble of order at most $k + 2$ costs $O(n^4r^2)$ time.