Beyond Classes of Graphs with "Few" Minimal Separators: FPT Results Through Potential Maximal Cliques

Mathieu Liedloff, Pedro Montealegre, and Ioan Todinca(✉)

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, Orléans, France
{mathieu.liedloff,pedro.montealegre,ioan.todinca}@univ-orleans.fr

Abstract. In many graph problems, like Longest Induced Path, Maximum Induced Forest, etc., we are given as input a graph G and the goal is to compute a largest induced subgraph $G[F]$, of treewidth at most a constant t, and satisfying some property $\mathcal{P}$. Fomin et al. [12] proved that this generic problem is polynomial on the class of graphs $\mathcal{G}_{\text{poly}}$, i.e., the graphs having at most $\text{poly}(n)$ minimal separators for some polynomial poly, when property $\mathcal{P}$ is expressible in counting monadic second order logic (CMSO).

Here we consider the class $\mathcal{G}_{\text{poly}} + kv$, formed by graphs of $\mathcal{G}_{\text{poly}}$ to which we may add a set of at most k vertices with arbitrary adjacencies, called *modulator*. We prove that the generic optimization problem is fixed parameter tractable on $\mathcal{G}_{\text{poly}} + kv$, with parameter k, if the modulator is also part of the input. The running time is of type $\mathcal{O}\left(f(k+t, \mathcal{P}) \cdot n^{t+5} \cdot (\text{poly}(n)^2)\right)$, for some function f.

1 Introduction

Many classical optimization problems on graphs, e.g., Maximum Independent Set, Maximum Induced Forest (whose optimal solution is the complement of a Minimum Feedback Vertex Set), Longest Induced Path and Maximum Induced Matching consist in finding a maximum induced subgraph $G[F]$ of the input graph G such that $G[F]$ has a tree-like structure (i.e., the treewidth is bounded by a constant) and satisfies some particular property $\mathcal{P}$ (like being a path, a matching, etc.). All these properties are expressible in Counting Monadic Second Order Logic (CMSO). We do not need in this paper the technical definition of CMSO formulae, for which the reader may refer to [10] or [12]. We only need to keep in mind that many natural properties (connectivity, excluding a fixed minor, etc.) are expressible in CMSO, and the fact that CMSO properties are *regular*, in a sense to be defined in the next section.

Fomin et al. [12] introduced the following generic optimization problem called Optimal Induced Subgraph for $\mathcal{P}$ and t, which encompasses those cited above and many others. In this generic problem, t is an integer constant and $\mathcal{P}$ is a property on graphs and vertex sets, expressible in CMSO.

E.W. Mayr (Ed.): WG 2015, LNCS 9224, pp. 499–512, 2016.
DOI: 10.1007/978-3-662-53174-7_35

Optimal Induced Subgraph for $\mathcal{P}$ and t

Input: A graph $G = (V, E)$
Output: A pair (F, X) of vertex subsets $X \subseteq F \subseteq V$ such that
- $\mathrm{tw}(G[F]) \leq t$,
- $\mathcal{P}(G[F], X)$ is true, and
- X is of maximum size under these constraints.

In the problems that we have mentioned, the vertex set X is equal to F. Nevertheless, set X allows to optimize other criteria than the size of the induced subgraph. E.g., in the Independent $\mathcal{H}$-packing problem [8], we are given a fixed family of graphs $\mathcal{H}$, and the goal is to find an induced subgraph $G[F]$ with a maximum number of connected components such that each of its components is isomorphic to an element of $\mathcal{H}$. For this problem, property $\mathcal{P}$ expresses the constraints on the components of $G[F]$ and the fact that X intersects each such component in exactly one vertex.

Consider a polynomial poly, and let $\mathcal{G}_{\text{poly}}$ be the class of graphs such that, for any $G \in \mathcal{G}_{\text{poly}}$, graph G has at most $\text{poly}(n)$ minimal separators. (As usually, we denote by n and m the number of vertices, respectively of edges of graph G.) Fomin et al. [12] proved that, for any constant t and any CMSO property $\mathcal{P}$, problem Optimal Induced Subgraph for $\mathcal{P}$ and t is polynomial-time solvable on class $\mathcal{G}_{\text{poly}}$.

The approach is based on the notion of *potential maximal clique*. Given an arbitrary graph $G = (V, E)$, a *minimal triangulation* $H = (V, F)$ is a minimal chordal supergraph of G (recall that a graph is chordal if it contains no induced cycle with four or more vertices). A *potential maximal clique* of G is a vertex subset Ω inducing a maximal clique in some minimal triangulation of G. Potential maximal cliques are strongly related to minimal separators. Graphs in $\mathcal{G}_{\text{poly}}$ have $\mathcal{O}(n \cdot (\text{poly}(n))^2)$ potential maximal cliques [7], and the set of all these objects can be enumerated in polynomial time. The algorithm of [12] takes as input all the potential maximal cliques of the input graph. Then it proceeds by dynamic programming on potential maximal cliques, constructs the induced subgraph $G[F]$ and in the meantime applies Courcelle's theorem [9], in the version proposed by Borie, Parker and Tovey [5], for testing CMSO properties on graphs on bounded treewidth. Altogether, this solves the generic optimization problem. Many graph classes, e.g., *weakly chordal*, *circle*, *polygon-circle* or *circular-arc* graphs are known to be in $\mathcal{G}_{\text{poly}}$ for some particular polynomial poly. Therefore, the generic problem and all its particular instances are polynomial on all these classes. We refer to [12] for further discussions on graph classes and applications of the problem.

Our results. Our goal is to study the problem from a parameterized perspective, for classes of graphs with "few" minimal separators, to which we are allowed to add k vertices with arbitrary adjacencies. Let $\mathcal{G}_{\text{poly}} + kv$ denote the class of graphs $G = (V, E)$ containing a vertex subset $M \subseteq V$ of size at most k, such that $G - M \in \mathcal{G}_{\text{poly}}$. The set M is called the *modulator* of G.

Let Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}} + kv$ be the problem Optimal Induced Subgraph for $\mathcal{P}$ and t on the graph class $\mathcal{G}_{\text{poly}} + kv$, with parameter k. *Moreover, we assume that the input graph G is given together with a modulator M* (see also Sect. 4 for discussions on this point). Our main result is that problem Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}} + kv$ is fixed-parameter tractable (FPT), i.e., there is an algorithm solving the problem in time $f(k) \cdot n^{O(1)}$ for some function f. More specifically, the running time is of type $\mathcal{O}\left(f(k+t,\mathcal{P}) \cdot n^{t+5} \cdot (\text{poly}(n)^2)\right)$, where function f depends on property $\mathcal{P}$ and on $k+t$ (see also the Conclusion section for further discussions).

Theorem 1. *Problem* Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}}+$ *kv with parameter k is fixed-parameter tractable, when a modulator is also part of the input.*

A similar result was obtained by Fomin and the authors of this article [11], on graphs with vertex cover at most k, i.e., formed by an independent set plus at most k vertices. In this class, the number of potential maximal cliques is $4^k \cdot n^{O(1)}$, hence the algorithm of [12] can be used as is. But, as shown in [11], even for the class of graphs formed by a tree plus one vertex, or an induced matching plus two vertices, the number of potential maximal cliques may be exponential in n, therefore we need another approach.

A natural idea is to “guess” how the optimal solution intersects with the modulator M. But we still need to carefully express how the rest of the solution intersects with the graph $G - M$. Think, e.g., of the longest induced path problem: we need to make sure that the solution restricted to $G - M$ forms indeed a connected path with the selected vertices of the modulator. Our algorithm extends, in a non-trivial way, the one of [12] in order to handle this situation.

We also point out that several authors considered classes of graphs of similar flavor, e.g. *Chordal* + *kv* [19] and *Split* + *kv* [18], providing both FPT and hardness results for various problems, parameterized by k.

2 Preliminaries

Treewidth, minimal triangulations and potential maximal cliques. Given a graph $G = (V, E)$, we denote by n the number of its vertices and by m the number of its edges. $G[C]$ denotes the subgraph of G induced by a vertex subset C, and $N(C)$ is the neighborhood of C in G. We say that a set of vertices C is a connected component of G if $G[C]$ is connected and C is inclusion-maximal for this property. Given a set $S \subseteq V$, let $G - S$ denote the graph $G[V \setminus S]$. If there are two distinct connected components C and D of $G - S$ such that $N(C) = N(D) = S$, we say that S is a *minimal separator* of G. The set of all minimal separators of G is denoted Δ_G.

A *tree decomposition* of graph $G = (V, E)$ is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{T}$ is a tree and $\mathcal{X}$ are vertex subsets of G, called *bags*. Moreover, each node i of the tree corresponds to a bag $X_i \in \mathcal{X}$, the bags cover all vertices and all edges of G, and for each vertex x of G, the set of nodes $\{i \mid x \in X_i\}$ form a connected subtree of $\mathcal{T}$. The *width* of the decomposition is $\max\{|X_i| - 1 \mid X_i \in \mathcal{X}\}$. Finally, the *treewidth* of G, denoted $\mathrm{tw}(G)$, is the minimum width among all mobile tree decompositions of G.

A graph H is *chordal* if it has no induced cycle with four or more vertices. Let $G = (V, E)$ be an arbitrary graph. We say that $H = (V, F)$ is a *triangulation* of G if H is a chordal super-graph of G (i.e., $E \subseteq F$). If, moreover, H is inclusion-minimal for this property, then H is a *minimal triangulation* of G. It is well-known that chordal graphs have tree decompositions whose bags correspond to maximal cliques (see, e.g., [15]). The treewidth of G is also equal to the minimum integer w such that G has a (minimal) triangulation H, and each clique of H has at most $w + 1$ vertices.

Proposition 1 (respecting triangulations [13]). *Consider an arbitrary graph $G = (V, E)$ and let $F \subseteq V$ be a set of vertices. For any minimal triangulation T_F of $G[F]$, there is a minimal triangulation T_G of G such that T_F is an induced subgraph of T_G.*

A *potential maximal clique* of G is a vertex subset Ω such that Ω induces a maximal clique in some minimal triangulation H of G. The set of all potential maximal cliques of G is denoted Π_G. E.g., if G is a cycle, then its potential maximal cliques are exactly the triples of vertices. See also [6] for a characterization of potential maximal cliques.

If Ω is a potential maximal clique, then the neighborhoods of the components of $G - \Omega$ are exactly the minimal separators of G, contained in Ω.

Given a polynomial poly, let $\mathcal{G}_{\text{poly}}$ denote the class of graphs having at most $\text{poly}(n)$ minimal separators. The minimal separators Δ_G and the potential maximal cliques Π_G of these graphs can be listed in polynomial time, by [1] and [7] respectively.

A pair (S, C) such that S is a minimal separator of G and C is a component such that $N(C) = S$ is called a *full block* associated to S. For convenience, we will also consider the empty set as being a minimal separator of $G = (V, E)$, and the pair $(\emptyset, V)$ is considered as a full block. Let (S, C) be a full block and Ω be a potential maximal clique with $S \subset \Omega \subseteq S \cup C$. The triple (S, C, Ω) is called a *good triple*. Our dynamic programming is based on full blocks and good triples, which is again polynomially bounded on class $\mathcal{G}_{\text{poly}}$.

Terminal recursive graphs and regular properties. Graphs of bounded treewidth can be defined recursively, based on a graph grammar. Let w be a non-negative integer. A *w-terminal graph* is a triple (V, T, E), where (V, E) is a graph and T is a *totally ordered* subset of V, of size at most w. The vertices of T are called the *terminals* of the graph. Since T is totally ordered, we can speak of the ith terminal, for $i \leq |T|$.

The class of *w-terminal recursive graphs* is defined by the following operations. A *base graph* is a w-terminal recursive graph of the form (V, T, E) with $T = V$. Hence it has at most w vertices, all of them being terminals.

The *gluing* operation takes two disjoint w-terminal recursive graphs $G_1 = (V_1, T_1, E_1)$ and $G_2 = (V_2, T_2, E_2)$ and creates a new graph $G = glue_m(G_1, G_2)$, depending on a matrix m. The matrix m has two rows and at most w columns, with elements in $\{0, 1 \ldots, w\}$. The gluing operation takes the disjoint union of G_1 and G_2, and then identifies the terminal number i in G_1 (resp. in G_2) to terminal m_{1i} (resp. m_{2i}) in G. Each terminal of G_1 (resp. G_2) is mapped on at most one terminal of G. We take $m_{ji} = 0$, for $j \in \{1, 2\}$, if terminal number i in G_j does not exist or it is not mapped on any terminal of G.

The *forget* operation takes a w-terminal recursive graph $G_1 = (V, T, E)$ and creates the graph $G = forget_m(G_1)$ with $G = (V, T', E)$ such that T' is a subset of T. The matrix m has only one row and $|T|$ columns, and m_{1i} specifies as before that the ith terminal of G_1 is mapped on terminal m_{1i} of G. The mapping is injective, and if $m_{1i} = 0$ then the ith terminal of G_1 is removed, in G, from the set of terminals.

We point out that the number of possible different matrices and hence of different operations is bounded by a function on w.

Proposition 2 (see [2,12]). *Graph $H = (V, T, E)$ is $(w + 1)$-terminal recursive if and only if there exists a tree decomposition of $G = (V, E)$, of width at most w, having T as one of its bags. Hence the grammar of $(w + 1)$-terminal recursive graphs constructs exactly the graphs of treewidth at most w (see [2,12]).*

Let $\mathcal{P}(G, X)$ be a property assigning to each graph G and vertex subset X of G a boolean value. We extend the gluing and forget operations to pairs (G, X) in the natural way (see, e.g., [5,12]). In particular, when we perform a gluing on (G_1, X_1) and (G_2, X_2), the result is a pair (G, X) where X is obtained by the the gluing of X_1 and X_2. Therefore the intersections of sets X_1 and X_2 with the terminals of G_1 and respectively G_2 must be coherent with the gluing, in the sense that if two terminals x_1 of G_1 and x_2 of G_2 are identified in G, then we either have $x_1 \in X_1$ and $x_2 \in X_2$, or we have $x_1 \notin X_1$ and $x_2 \notin X_2$.

Definition 1 (regular property). *Property $\mathcal{P}$ is called* regular *if, for any value w, we can associate a finite set $\mathcal{C}$ of* classes *and a* homomorphism function *h, assigning to each w-terminal recursive graph G and to each vertex subset X a class $h(G, X) \in \mathcal{C}$ such that:*

1. *If $h(G_1, X_1) = h(G_2, X_2)$ then $\mathcal{P}(G_1, X_1) = \mathcal{P}(G_2, X_2)$.*
2. *For each gluing operation $glue_m$ there exists a function $\odot_{glue_m} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ such that, for any two pairs (G_1, X_1) and (G_2, X_2),*

$$h(glue_m((G_1, X_1), (G_2, X_2)) = \odot_{glue_m}(h(G_1, X_1), h(G_2, X_2))$$

and for each operation $forget_m$ there is a function $\odot_{forget_m} : \mathcal{C} \to \mathcal{C}$ such that, for any pair (G, X),

$$h(forget_m(G, X)) = \odot_{forget_m}(h(G, X)).$$

The first condition separates the classes into *accepting* ones (i.e., classes $c \in \mathcal{C}$ such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is true) and *rejecting* ones (s.t. $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is false). In full words, the second condition states that, if we perform a *glue* (resp. *forget*) operation on two graphs (resp. one graph) and corresponding vertex subsets, the homomorphism class of the result can be obtained from the homomorphism classes of the graphs on which these operations are applied. Therefore, if a w-terminal recursive graph is given together with its expression in this grammar, and if moreover we know how to compute the classes of the base graph, then the homomorphism class of the whole graph, for a regular property $\mathcal{P}$, can be obtained by dynamic programming. We simply need to parse the expression from bottom to top and, at each node, we compute the class of the corresponding sub-expression thanks to the second condition of regularity. At the root, the property is true if and only if we are in an accepting class.

Proposition 3 (Borie, Parker, and Tovey [5], **Courcelle** [9]**).** *Any property $\mathcal{P}(G, X)$ expressible by a CMSO formula is regular.*

Moreover, the result of Borie, Parker, and Tovey shows how to compute explicitly the set of classes, the homomorphism function for base graphs as well as the composition functions $\odot_{glue_m}$ and $\odot_{forget_m}$. Altogether, this provides an effective algorithm for checking the property in $O(n)$ time.

The reader may try to express the homomorphism classes and function for his/her favorite CMSO property. Let us consider the property "$G[X]$ is connected". We can choose, as homomorphism $h((V,T,E),X)$, the set of subsets of T, which correspond to the intersections of T with components of $G[X]$. Observe that each such subset T_i of T is encoded by the indices of its elements in the ordered set T. Hence each homomorphism class will be a set of (disjoint) subsets of $\{1,\ldots,w\}$.

We may assume w.l.o.g. that the homomorphism class $c = h(G,X)$, for $G = (V,T,E)$, encodes the intersection of X with the set of terminals. This is not explicitly required by the definition of regular properties, but it can be done since it only costs w bits to encode the number of the terminals contained in X. Therefore we assume there is a function $trm(c,T)$ that, given a homomorphism class c and an ordered set of terminals T returns the unique possible set $X \cap T$, over all pairs $(G = (V,T,E), X)$ mapped to c. Thanks to this function, when we will glue two terminal recursive graphs with their corresponding vertex subsets, we will be able to check that the gluing is coherent.

3 The Algorithm

Our goal is to provide an FPT algorithm for the problem Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}} + kv$, thus proving our Main Theorem 1. Recall that in this problem, the input is a graph $G \in \mathcal{G}_{\text{poly}} + kv$, together with a modulator M of size at most k.

We may assume w.l.o.g. that we also have as input the set $\Pi_{G'}$ of potential maximal cliques of graph $G' = G - M$. Indeed these objects can be computed in polynomial time by [7].

The following easy observation is crucial for the correctness of our algorithm.

Lemma 1 (compatibility lemma). *Let G be the input graph, M be a vertex subset, and (F, X) be an optimal solution for $\mathcal{P}$ and t. Let $G' = G - M$ and $F' = F \setminus M$. There is a minimal triangulation $T_{F'}$ of $G'[F']$ of width at most t, and a minimal triangulation $T_{G'}$ of G' respecting $T_{F'}$, i.e., such that $T_{F'}$ is the subgraph induced by F' in $T_{G'}$.*

Proof. Since $G[F]$ is of treewidth at most t, so is its subgraph $G'[F']$. Therefore it exists a minimal triangulation $T_{F'}$ of $G'[F']$ of width at most t. By Proposition 1 applied to graph G' and to $T_{F'}$, there is a minimal triangulation $T_{G'}$ of G', respecting the minimal triangulation $T_{F'}$. □

We will “guess”, by brute force, the intersections of F and X with the modulator M. Let us fix $F^M = F \cap M$ and $X^M = X \cap M$, with $X^M \subseteq F^M$. For each such pair (F^M, X^M), we need to construct $F' = F \setminus M$ and $X' = X \setminus M$ such that the pair $(F, X) = (F' \cup F^M, X' \cup X^M)$ satisfies the constraints and $X' \cup X^M$ is of maximum size under these conditions. (Eventually, the global solution is obtained by trying all the 3^k possible combinations for subsets $X^M \subseteq F^M \subseteq M$.)

Our algorithm will construct X' and F' by dynamic programming on minimal separators and potential maximal cliques. The graph $G[F' \cup F^M]$ has to be of treewidth at most t, and while we construct this graph we also need to check property $\mathcal{P}$ on it. Unfortunately we will not be able to handle $G[F' \cup F^M]$ as a $(t+1)$-terminal recursive graph. Instead, we will see it as a $(t+k^M+1)$-terminal recursive graph, where $k^M = |F^M|$ (and we will explicitly check that $\text{tw}(G[F]) \leq t$). Informally, while we construct F', we also maintain some information on a tree-decomposition of $G'[F']$, of width at most t. To this decomposition, we simply add the whole set F^M in each bag, hence obtaining a tree-decomposition of width at most $t + k^M$ of $G[F \cup F^M]$.

Now, on the $(t + k^M + 1)$-terminal recursive graph $G[F' \cup F^M]$ having $W \cup F^M$ as set of terminals for some W (W will be memorized during the dynamic programming), we need to check the property $\mathcal{P}$ but also the fact that the graph is of treewidth at most t. Therefore, let $\mathcal{Q}(H, Y)$ be the property $\mathcal{P}(H, Y) \wedge (\text{tw}(H) \leq t)$. Property $\mathcal{Q}$ is also regular:

Lemma 2. *Let $\mathcal{P}(G, X)$ be a regular property on graphs and vertex sets, and let $\mathcal{Q}(G, X)$ be the property $\mathcal{P}(G, X) \wedge (\text{tw}(G) \leq t)$. Property $\mathcal{Q}$ is regular.*

Proof. As proven by Borie, Parker, and Tovey [5], if two properties $\mathcal{P}(G, X)$ and $\mathcal{P}'(G)$ are regular, then property $\mathcal{Q}(G, X) = \mathcal{P}(G, X) \wedge \mathcal{P}'(G)$ is also regular. In order to observe this, one can simply notice that the couple $(h_{\mathcal{P}}(G, X), h_{\mathcal{P}'}(G))$ formed by the respective classes of $\mathcal{P}$ and $\mathcal{P}'$ directly provides the homomorphism class $h_{\mathcal{Q}}(G, X)$ for property $\mathcal{Q}$.

It remains to argue that property $\mathcal{P}'(G)$ defined by “$\text{tw}(G) \leq t$” is regular. One classical argument is that the class of graphs of treewidth at most t is minor-closed (see, e.g., [3] for a similar discussion). Hence, by the Graph Minors theorem [21], the class is defined by a finite set of forbidden minors, denoted $Obs(t)$. Therefore, $\text{tw}(G) \leq t$ if and only if G has no minor among the graphs of $Obs(t)$. The property that a fixed graph is a minor of G is expressible in CMSO (see, e.g., [5]), hence the property "$\text{tw}(G) \leq t$" is regular by Proposition 3. In order to turn this argument into a completely constructive one, we must build the obstruction set $Obs(t)$ for graphs of treewidth at most t. This can be done by brute force, thanks to the result of Lagergren [17] showing that such obstructions are of size (number of vertices) at most $f(t)$, for some function doubly exponential in t^5. Hence, one could enumerate all the graphs with number of vertices bounded by this function, test the ones of treewidth strictly larger than t and extract the minimal ones w.r.t. the minor relation. The output is precisely $Obs(t)$.

For our purpose, a better alternative is provided by the celebrated algorithm of Bodlaender and Kloks [4]. This algorithm takes as input a graph of treewidth at most w, for some constant w, and decides if this graph has treewidth at most t. Its running time is $O(n)$, and the hidden constant is single exponential in a polynomial in w. Moreover, the algorithm precisely provides an effective way for constructing the homomorphism class of property $\mathcal{P}'(G) = (\text{tw}(G) \leq t)$ for $(w+1)$-terminal recursive graphs. Such a class is addressed in [4] as a *full set of characteristics*. Recall that, in our case, we need to construct these class for $(w+1)$-terminal recursive graphs where $w \leq t+k$, k being the size of the modulator M. Therefore, for computing the homomorphism classes for property $\mathcal{Q}(G,X) = \mathcal{P}(G,X) \wedge (\text{tw}(G) \leq t)$ one can combine the Borie, Parker, and Tovey approach (to obtain $h_{\mathcal{P}}(G,X)$) and the Bodlaender and Kloks approach (to obtain $h_{\mathcal{P}'}(G)$ for $\mathcal{P}'(G) = (\text{tw}(G) \leq t)$), and the couple $(h_{\mathcal{P}}(G,X), h_{\mathcal{P}'}(G))$ is the homomorphism class $h_{\mathcal{Q}}(G,X)$. □

Our algorithm is an extension of the dynamic programming scheme proposed by Fomin et al. [12], which considers the same optimization problem, without the modulator M (and thus checking property $\mathcal{P}$ instead of $\mathcal{Q}$). For a better understanding we completely describe the new algorithm, trying to follow the same notations as in [12], and we emphasize the points that differ from the result of Fomin et al.

Recall that sets X^M and F^M are fixed, and $X^M \subseteq F^M \subseteq M$. We consider a total order $(v_1, \ldots, v_n)$ on the vertices of $G = (V,E)$. When we speak of a subset T of vertices as a set of terminals, T is considered as the ordered set, with the ordered induced by $(v_1, \ldots, v_n)$ on its vertices.

Definition 2 (partial compatible solution). *Consider a full block (S,C) and a good triple (S,C,Ω) of graph G'. Let $W \subseteq S$ (resp. $W \subseteq \Omega$) a vertex set of size at most $t+1$. Let c be a homomorphism class for property $\mathcal{Q}$ on $(t+k^M+1)$-terminal recursive graphs. We say that a pair (F,X) is a* partial solution compatible with (S,C,W,c) *(resp. with (S,C,Ω,W,c)) if the following conditions hold:*

1. *$X \cap M = X^M$, $F \cap M = F^M$, and $X \subseteq F$.*
2. *$F \setminus M \subseteq S \cup C$ and $F \cap S = W$ (resp. $F \cap \Omega = W$).*
3. *$H = (F, W \cup F^M, E(G[F]))$ is a $(t+k^M+1)$-terminal recursive graph, and the homomorphism class $h_{\mathcal{Q}}(H,X)$ for property Q is exactly c.*

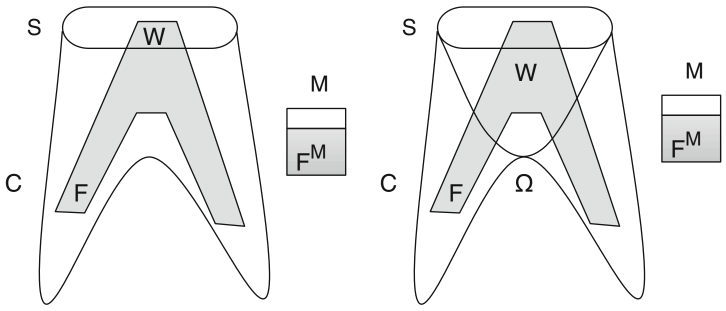

Fig. 1. (a) Partial solutions compatible with (S, C, W, c) (left), and (b) with (S, C, Ω, W, c) (right). Set F is depicted in grey. Note that set W corresponds to $F \cap S$ in the first case, and to $F \cap \Omega$ in the second case.

4. There is a minimal triangulation $T_{F'}$ of $G'[F']$ (here $F' = F \setminus M$) and a minimal triangulation $T_{G'}$ of G' respecting $T_{F'}$, such that S is a minimal separator (resp. Ω is a maximal clique) of $T_{G'}$.

With the same notations as above, let $\alpha(S, C, W, c)$ (resp. $\beta(S, C, \Omega, W, c)$) be the maximum size of X over all partial solutions (F, X) compatible with (S, C, W, c) (resp. (S, C, Ω, W, c)). The situation is depicted in Fig. 1. For simplicity, we did not represent set X. The algorithm orders the full blocks (S, C) by the size of $S \cup C$. It proceeds by dynamic programming on full blocks (S, C) in this order, and on good triples (S, C, Ω), computing all possible values $\alpha(S, C, W, c)$ and $\beta(S, C, \Omega, W, c)$. The outline of Algorithm 1 is the same as in [12], the differences appear in the details of the computations of α and β values.

Algorithm 1. Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}} + kv$

Input: graph $G = (V, E)$ and a modulator M of size at most k s.t. $G' = G - M$ is in $\mathcal{G}_{\text{poly}}$; the potential maximal cliques of G'; sets $X^M \subseteq F^M \subseteq M$

Output: sets $X \subseteq F \subseteq V(G)$ such that $G[F]$ has treewidth at most t, $\mathcal{P}(G[F], X)$ is true, $X \cap M = X^M$, $F \cap M = F^M$ and, subject to these constrains, X is of maximum size

```
1  Order all full blocks $(S, C)$ of $G'$ by inclusion on $S \cup C$;
2  for all full blocks $(S, C)$ in this order do
3      for all good triples $(S, C, \Omega)$ of $G'$, all $W \subseteq \Omega$ of size $\leq t+1$ and all $c \in \mathcal{C}$ do
4          if $\Omega = S \cup C$ then Compute $\beta(S, C, \Omega, W, c)$ using Eq. 1;
5          ;
6          else Compute $\beta(S, C, \Omega, W, c)$ using Eqs. 3, 4, 5, and 6;
7          ;
8      for all $W \subseteq S$ of size $\leq t+1$ and all $c \in \mathcal{C}$ do
9          Compute $\alpha(S, C, W, c)$ using Eq. 2;
10 Compute an optimal solution using Eq. 7;
```

Base case: the good triple (S, C, Ω) *is such that* $\Omega = S \cup C$. In this case the only possible partial solutions (F, X) compatible with (S, C, Ω, W, c) correspond to base graphs $G[F]$, where all vertices are terminals (see [12]). Hence $F = W \cup F^M$ is also the set of terminals. Thus set X is unique (or might not exist), because we must have $X = trm(c, W \cup F^M)$. For simplicity, we denote by $G[W \cup F^M]$ the base graph $(W \cup F^M, W \cup F^M, E(G[F \cup F^M]))$. We have:

$$\beta(S, C, \Omega, W, c) = \begin{cases} |X| & \text{if there is } X \subseteq W \text{ such that } h(G[W \cup F^M], X) = c \\ -\infty & \text{otherwise} \end{cases} \tag{1}$$

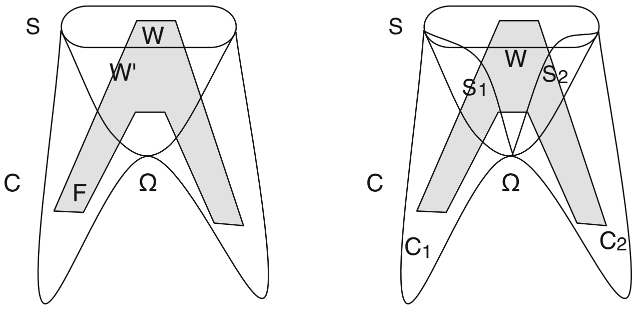

Fig. 2. Computing α form β (left), and β from α (right). Set M is not depicted for simplicity.

Computing α from β. We aim to compute $\alpha(S, C, W, c)$ from β values on good triples of type (S, C, Ω) (see also Fig. 2(a)).

Let (F, X) be an optimal solution compatible with (S, C, W, c) and $F' = F \setminus M$. We denote by H the $(t + k^M + 1)$-terminal recursive graph $G[F]$ with $W \cup F^M$ as set of terminals. By Definition 2, there is a minimal triangulation $T_{F'}$ of F' and a minimal triangulation $T_{G'}$ of G' respecting $T_{F'}$, such that S is a minimal separator of $T_{G'}$. By [6], there is a potential maximal clique Ω of G', inducing a maximal clique in $T_{G'}$, and such that (S, C, Ω) form a good triple of G'. Let $W' = F' \cap \Omega$. The graph H', corresponding to $G[F]$ with set of terminals $W' \cup F^M$, is also a $(t+k^M+1)$-terminal recursive graph (see Proposition 2, or [12] for full details). Let $c' = h(H', X)$. Note that $H = forget_{W' \cup F^M \to W \cup F^M}(H)$, where the *forget* operation corresponds to the fact that the set of terminals $W' \cup F^M$ is reduced to $W \cup F^M$. Therefore, we have (see [12] for full details):

$$\alpha(S, C, W, c) = \max \beta(S, C, \Omega, W', c'), \tag{2}$$

where the maximum is taken over potential maximal cliques Ω such that (S, C, Ω) is a good triple, all subsets $W' \subseteq \Omega$ of size at most $t + 1$ such that $W' \cap S = W$ and all classes $c' \in \mathcal{C}$ such that $\odot_{forget_{W' \cup F^M \to W \cup F^M}}(c') = c$.

Computing β from α. Let (S, C, Ω) be a good triple of G. Denote by $C_1, \ldots, C_p$ the components of $G' - \Omega$ contained in C, and let S_i be the neighborhood of C_i in G'. The pairs (S_i, C_i) are also full blocks (see [6] for more details) and they have been processed by the algorithm before (S, C). Our goal is to compute $\beta(S, C, \Omega, W, c)$. Let $W_i = W \cap S_i$, for all $i \in \{1, \ldots p\}$. We will use, as in [12], two intermediate functions γ_i and δ_i.

Let $\delta_i(S, C, \Omega, W, c_i^+)$ denote $\max |X_i|$ over the partial solutions (F_i^+, X_i) compatible with (S, C, Ω, W, c_i^+) and such that $F_i^+ \setminus F^M \subseteq \Omega \cap C_i$. Let H_i^+ be the graph $G[F_i^+]$ with set of terminals $W \cup F^M$. Let also H_i be the graph $H_i^+[S_i \cup C_i \cup M]$, with set of terminals $W_i \cup F^M$. Note that H_i^+ is obtained by gluing H_i with the base graph $G[W \cup F^M]$, with the “canonical” gluing, obtained by identifying the vertices of S_i from both sides. Let $glue_{W_i \cup F^M; W \cup F^M}$ denote this gluing operation.

$$\delta_i(S, C, \Omega, W, c_i^+) = \max \alpha(S_i, C_i, W_i, c_i) + |trm(c_W, W \cup F^M) \setminus trm(c_i, W_i \cup F^M)|, \tag{3}$$

where the maximum is taken over all $c_i, c_W \in \mathcal{C}$ s.t. $\odot_{glue_{W_i \cup F^M; W \cup F^M}}(c_i, c_W) = c_i^+$ and $c_W = h(G[W \cup F^M], X_W \cup X^M)$ for some $X_W \subseteq W$. Here $G[W \cup F^M]$ denotes the base graph with terminals $W \cup F^M$.

Notice the part $|trm(c_W, W \cup F^M) \setminus trm(c_i, W_i \cup F^M)|$ in the formula, which avoids the overcounting of the vertices of $X_i \cap S_i$.

These partial solutions (F_i^+, X_i^+) corresponding to $\delta_i(S, C, \Omega, W, c_i^+)$ cannot be glued together in one step, since we are only allowed to glue two graphs at a time. Hence the need of the γ function which allows to add, one by one, the partial solutions to the gluing. Now let $\gamma_i(S, C, \Omega, W, c)$ denote the size of the optimal partial solution compatible with (S, C, Ω, W, c) and contained in $M \cup \Omega \cup C_1 \cdots \cup C_i$. So we only consider the first i components, the partial solution is the union of (F_1^+, X_1^+) to (F_i^+, X_i^+). By definition,

$$\gamma_1(S, C, \Omega, W, c) = \delta_1(S, \Omega, C, W, c) \tag{4}$$

We then compute γ_i, for i from 2 to p as follows.

$$\gamma_i(S, C, \Omega, W, c) = \max \gamma_{i-1}(S, C, \Omega, W, c') + \delta_i(S, \Omega, C, W, c'') - |trm(c', W \cup F^M)|, \tag{5}$$

where the maximum is taken over all $c', c'' \in \mathcal{C}$ s. t. $\odot_{glue_{W \cup F^M; W \cup F^M}}(c', c'') = c$, where the gluing operation is the canonical gluing, the set of terminals for both arguments being $W \cup F^M$.

By definition of γ_p, we have

$$\beta(S, C, \Omega, W, c) = \gamma_p(S, C, \Omega, W, c). \tag{6}$$

It remains to retrieve the optimal solution for the algorithm. The maximum is taken over all accepting classes c, i.e., classes such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$:

$$\max \alpha(\emptyset, V, \emptyset, c), \tag{7}$$

We refer to [12] for detailed proofs of correctness and for complexity issues. Altogether, the algorithm takes time $\mathcal{O}(f(t+k^M,\mathcal{P})\cdot n^{t+4}\cdot|\Pi_{G'}|)$. The function $f(t+k^M,\mathcal{P})$ comes from the application of Proposition 3 on $t+k^M+1$-recursive graphs. By applying Algorithm 1 on all possible subsets $X^M\subseteq F^M\subseteq F$, we have proved Theorem 1.

We point out that our algorithm really needs to keep track of the homomorphism classes of partial solutions (F,X) for property $\mathcal{Q}(G[F],X)=\mathcal{P}(G[F],X)\wedge(\text{tw}(G[F])\leq t)$. A naïve approach would be to only keep the class $h_{\mathcal{P}}(G[F],X)$ (for property $\mathcal{P}$) and to reject partial solutions that do not satisfy $\text{tw}(G[F])\leq t$. We could have two different partial solutions (F,X) and (F',X) for the same part of the graph (e.g., corresponding to the same parameters for function α), such that $h_{\mathcal{P}}(G[F],X)=h_{\mathcal{P}}(G[F'],X')$, and both $\text{tw}(G[F])$ and $\text{tw}(G[F'])$ are at most t. Or, it may happen that one of the solution, say (F,X), can be extended into a better one of type $(F\cup F'',X\cup X'')$, while the other cannot because such an extension$(F'\cup F'',X'\cup X'')$ would violate the condition $\text{tw}(G[F'\cup F''])\leq t$.

Of course our approach, keeping the class $h_{\mathcal{Q}}(G[F],X)$ for property $\mathcal{Q}$, ensures that if two partial solutions are of the same class, they are equivalent w.r.t. extensions.

4 Conclusion and Discussion

We gave an FPT algorithm for the problem Optimal Induced Subgraph for $\mathcal{P}$ and t on $\mathcal{G}_{\text{poly}}+kv$. The problem encompasses many classical ones [12]. As it will be shown in the full version of the paper, the result can be extended to the classes of graphs $\mathcal{G}_{\text{poly}}-ke$ and $\mathcal{G}_{\text{poly}}+ke$ (i.e., graphs of $\mathcal{G}_{\text{poly}}$ minus or plus ar most k edges), and the generic problem is polynomial on $\mathcal{G}_{\text{poly}}-kv$, if k is small. One of the limits of our algorithm is that we explicitly need the modulator of the input graph. Let us consider the following problem:

Deletion to $\mathcal{G}_{\text{poly}}$
Input: A graph $G=(V,E)$ and a polynomial poly
Parameter: k
Output: A vertex subset M of size at most k, such that $G-M$ is in $\mathcal{G}_{\text{poly}}$

Our main open question is the existence of an FPT algorithm for problem Deletion to $\mathcal{G}_{\text{poly}}$. We recall that the problem Chordal deletion is FPT [20], but on the other hand the problem Weakly chordal deletion is $W[2]$-hard [16]. The latter does not rule out the possibility that Deletion to $\mathcal{G}_{\text{poly}}$ could be FPT. Moreover, even an FPT approximation for Deletion to $\mathcal{G}_{\text{poly}}$ would allow us to conclude that the problem Optimal Induced Subgraph for $\mathcal{P}$ and t is FPT on the class $\mathcal{G}_{\text{poly}}+kv$, without needing to require to have the modulator M as part of the input.

Another direction for improvement concerns the complexity of our algorithm, which is $\mathcal{O}\left(f(k+t,\mathcal{P})\cdot n^{t+5}\cdot(\text{poly}(n)^2)\right)$. The dependency on $\mathcal{P}$ and $t+k$ comes from Courcelle's theorem (Proposition 3), applied for deciding property $\mathcal{P}$ on graphs of treewidth $t + k$. As shown by Frick and Grohe [14], function f can be very huge, typically a tower of exponentials in $t + k$, the height of the tower depending on the property to be checked. Our algorithm actually constructs an induced subgraph of treewidth t, although we were only able to build a decomposition of width $t + k$. In particular, if we do not need to check a particular property $\mathcal{P}$ on the induced graph, but we only ask this graph to be of treewidth at most t, then function f becomes $(k+t)^{O((k+t)^3)}$ — coming from the algorithm of Bodlaender and Kloks [4] that, given a graph and tree decomposition of width $k + t$, checks whether the treewidth of the graph is at most t. For easier cases, when $t = 0$ to $t = 1$, the function f becomes 2^k and $(k + t)^{O(k+t)}$ respectively, as we shall discuss in the full version. Also, for natural properties $\mathcal{P}$, like "being connected" or "being a path" one can perform the property checking using standard (ad-hoc) dynamic programming tools which avoid the heavy machinery of Proposition 3. Again, the extra-cost becomes much more reasonable.

A natural and challenging question would be to separate the dependency on t and k, typically to obtain a complexity of type $f(t, \mathcal{P}) \cdot g(k) \cdot n^{t+\mathcal{O}(1)}$, where g would be a "more reasonable" function. For that purpose we would need to construct the partial solutions as a $(t+1)$-terminal recursive graph, maybe by a more clever way of dealing with the intersection between this solution and the modulator.

Acknowledgements. We would like to thank Fedor Fomin and Nicolas Nisse for fruitful discussions on this subject.

References

1. Berry, A., Bordat, J.P., Cogis, O.: Generating all the minimal separators of a graph. Int. J. Found. Comput. Sci. **11**(3), 397–403 (2000)
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**(1–2), 1–45 (1998)
3. Bodlaender, H.L.: Fixed-parameter tractability of treewidth and pathwidth. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) The Multivariate Algorithmic Revolution and Beyond. LNCS, vol. 7370, pp. 196–227. Springer, Heidelberg (2012)
4. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms **21**(2), 358–402 (1996)
5. Borie, R.B., Gary Parker, R., Tovey, C.A.: Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. Algorithmica **7**(5–6), 555–581 (1992)
6. Bouchitté, V., Todinca, I.: Treewidth and minimum fill-in: grouping the minimal separators. SIAM J. Comput. **31**(1), 212–232 (2001)
7. Bouchitté, V., Todinca, I.: Listing all potential maximal cliques of a graph. Theor. Comput. Sci. **276**(1–2), 17–32 (2002)
8. Cameron, K., Hell, P.: Independent packings in structured graphs. Math. Program. **105**(2–3), 201–213 (2006)
9. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Inf. Comput. **85**(1), 12–75 (1990)

10. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic. Cambridge University Press, Cambridge (2012)
11. Fomin, F.V., Liedloff, M., Montealegre, P., Todinca, I.: Algorithms parameterized by vertex cover and modular width, through potential maximal cliques. In: Ravi, R., Gørtz, I.L. (eds.) SWAT 2014. LNCS, vol. 8503, pp. 182–193. Springer, Heidelberg (2014)
12. Fomin, F.V., Todinca, I., Villanger, Y.: Large induced subgraphs via triangulations and CMSO. SIAM J. Comput. **44**(1), 54–87 (2015)
13. Fomin, F.V., Villanger, Y.: Finding induced subgraphs via minimal triangulations. In: STACS 2010, LIPIcs, pp. 383–394. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
14. Frick, M., Grohe, M.: The complexity of first-order and monadic second-order logic revisited. Ann. Pure Appl. Logic **130**(1–3), 3–31 (2004)
15. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)
16. Heggernes, P., van't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. Theor. Comput. Sci. **511**, 172–180 (2013)
17. Lagergren, J.: Upper bounds on the size of obstructions and intertwines. J. Comb. Theor. Ser. B **73**(1), 7–40 (1998)
18. Mancini, F.: Minimum fill-in and treewidth of split+ke and split+kv graphs. Discrete Appl. Math. **158**(7), 747–754 (2010)
19. Marx, D.: Parameterized coloring problems on chordal graphs. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 83–95. Springer, Heidelberg (2004)
20. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica **57**(4), 747–768 (2010)
21. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner's conjecture. J. Comb. Theor. Ser. B **92**(2), 325–357 (2004). Special Issue Dedicated to Professor W.T. Tutte