HAL
open science

P-FPT algorithms for bounded clique-width graphs

David Coudert, Guillaume Ducoffe, Alexandru Popa

To cite this version:

David Coudert, Guillaume Ducoffe, Alexandru Popa. P-FPT algorithms for bounded clique-width graphs. ACM Transactions on Algorithms, 2019, 15 (3), pp.1-57. 10.1145/3310228 . hal-02152971

HAL Id: hal-02152971

https://hal.science/hal-02152971

Submitted on 11 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

P-FPT algorithms for bounded clique-width graphs *

David Coudert[1], Guillaume Ducoffe[1,2,3], and Alexandru Popa[2,4]

[1]Université Côte d'Azur, Inria, CNRS, I3S, France
[2]National Institute for Research and Development in Informatics, Romania
[3]The Research Institute of the University of Bucharest ICUB, Romania
[4]University of Bucharest, Faculty of Mathematics and Computer Science

Abstract

Recently, hardness results for problems in P were achieved using reasonable complexity theoretic assumptions such as the Strong Exponential Time Hypothesis. According to these assumptions, many graph theoretic problems do not admit truly subquadratic algorithms. A central technique used to tackle the difficulty of the above mentioned problems is fixed-parameter algorithms with *polynomial dependency* in the fixed parameter (P-FPT). Applying this technique to *clique-width*, an important graph parameter, remained to be done.

In this paper we study several graph theoretic problems for which hardness results exist such as *cycle problems*, *distance problems* and *maximum matching*. We give hardness results and P-FPT algorithms, using clique-width and some of its upper bounds as parameters. We believe that our most important result is an algorithm in $\mathcal{O}(k^4 \cdot n + m)$-time for computing a maximum matching where k is either the modular-width of the graph or the P_4-sparseness. The latter generalizes many algorithms that have been introduced so far for specific subclasses such as cographs. Our algorithms are based on preprocessing methods using modular decomposition and split decomposition. Thus they can also be generalized to some graph classes with unbounded clique-width.

1 Introduction

The classification of problems according to their complexity is one of the main goals in computer science. This goal was partly achieved by the theory of NP-completeness which helps to identify the problems that are unlikely to have polynomial-time algorithms. However, there are still many problems in P for which it is not known if the running time of the best current algorithms can be improved. Such problems arise in various domains such as computational geometry, string matching or graph theory. In this paper, we focus on the existence and the design of *linear-time* algorithms, for solving several graph problems when restricted to classes of bounded *clique-width*. The problems considered comprise the detection of short cycles (*e.g.*, Girth and Triangle Counting), some distance problems (*e.g.*, Diameter, Hyperbolicity, Betweenness Centrality) and the computation of maximum matchings in graphs. We refer to Sections 3.1, 4.1 and 5, respectively, for a recall of their definitions.

*Results of this paper were partially presented at the SODA'18 conference [29].

Clique-width is an important graph parameter in structural graph theory, that intuitively represents the closeness of a graph to a cograph — *a.k.a.*, P_4-free graphs [26, 35]. Some classes of perfect graphs including distance-hereditary graphs and so, trees, have bounded clique-width [68]. Furthermore, clique-width has many algorithmic applications. Many algorithmic schemes and metatheorems have been proposed for classes of bounded clique-width [34, 31, 46]. – We refer to [38] for a more general discussion to parameterized complexity. – Perhaps the most famous result in this area is Courcelle's theorem, that states that every graph problem expressible in Monadic Second Order logic (MSO_1) can be solved in $f(k) \cdot n$-time when restricted to graphs with clique-width at most k, for some computable function f that only depends on k [34]. Some of the problems considered in this work can be expressed as an MSO_1 formula. However, the dependency on the clique-width in Courcelle's theorem is super-polynomial, that makes it less interesting for the study of graphs problems in P. Our goal is to derive a *fine-grained* complexity of polynomial-time solvable graph problems when restricted to classes of bounded clique-width, that requires different tools than Courcelle's theorem. – We stress that in sharp contrast to polynomial-time solvable problems, the fine-grained complexity of *NP-hard* problems when parameterized by clique-width is now rather well-understood [50, 51, 52]. –

Our starting point is the recent theory of "Hardness in P" that aims at better hierarchizing the complexity of polynomial-time solvable problems [99]. This approach mimics the theory of NP-completeness. Precisely, since it is difficult to obtain unconditional hardness results, it is natural to obtain hardness results assuming some complexity theoretic conjectures. In other words, there are key problems that are widely believed not to admit better algorithms such as 3-SAT (k-SAT), 3SUM and All-Pairs Shortest Paths (APSP). Roughly, a problem in P is hard if the existence of a faster algorithm for this problem implies the existence of a faster algorithm for one of these fundamental problems mentioned above. In their seminal work, Vassilevska Williams and Williams [100] prove that many important problems in graph theory are all equivalent under subcubic time reductions. That is, if one of these problems admits a truly subcubic time algorithms, then all of them do. Their results have extended and formalized prior work from, *e.g.*, [61, 79]. The list of such problems was further extended by Abboud et al. [1] and Borassi et al [18].

Besides purely negative results (*i.e.*, conditional lower bounds) the theory of "Hardness in P" also comes with renewed algorithmic tools in order to leverage the existence, or the nonexistence, of improved algorithms for some graph classes. The tools used to improve the running time of the above mentioned problems are similar to the ones used to tackle NP-hard problems, namely approximation and FPT algorithms. Our work is an example of the latter, of which we first survey the most recent results.

1.1 Related work: Fully polynomial parameterized algorithms.

FPT algorithms for polynomial-time solvable problems (or P-FPT algorithms for short) were first considered by Giannopoulou et al. [66]. Such a parameterized approach makes sense for any problem in P for which a conditional hardness result is proved, or simply no linear-time algorithms are known. Interestingly, the authors of [66] proved that a matching of cardinality at least μ in a graph can be computed in $\mathcal{O}(\mu n + \mu^3)$-time. We stress that Maximum Matching is a classical and intensively studied problem in computer science [41, 54, 55, 59, 78, 85, 84, 101]. The well known algorithm in [85], that runs in $\mathcal{O}(m\sqrt{n})$-time, is essentially the best so far for Maximum Matching. Approximate solutions were proposed by Duan and Pettie [41].

More related to our work is the seminal paper of Abboud, Vassilevska Williams and Wang [3]. They obtained rather surprising results when using *treewidth*, another important structural graph parameter that intuitively measures the closeness of a graph to a tree [14]. Treewidth has tremendous applications in pure graph theory [93] and parameterized complexity [30]. Furthermore, improved algorithms have long been known for "hard" graph problems in P, such as Diameter and Maximum Matching, when restricted to trees [77]. However, it has been shown in [3] that under the Strong Exponential Time Hypothesis, for any $\varepsilon > 0$ there can be no $2^{o(w)} \cdot n^{2-\varepsilon}$-time algorithm for computing the diameter of graphs with treewidth at most w. This hardness result even holds for *pathwidth*, that leaves little chance to find an improved algorithm for any interesting subclass of bounded-treewidth graphs while avoiding an exponential blow-up in the parameter. We show that the situation is different for clique-width than for treewidth, in the sense that the hardness results for clique-width do not hold for important subclasses.

We want to stress that a familiar reader could ask why the hardness results above do not apply to clique-width directly since it is upper bounded by a function of treewidth [27]. However, clique-width cannot be *polynomially* upper bounded by the treewidth [27]. Thus, the combination of the hardness results from [3] with the classical relationship between treewidth and clique-width do not preclude the existence of, say, an $\mathcal{O}(kn)$-time algorithm for computing the diameter of graphs with clique-width at most k. In order to prove such an impossibility result, we need to use other relationships between the two parameters.

On a more positive side, the authors in [3] show that Radius and Diameter can be solved in time $2^{\mathcal{O}(w \log w)} \cdot n^{1+o(1)}$, where w is treewidth. Husfeldt [73] shows that the eccentricity of every vertex in an undirected graph on n vertices can be computed in time $n \cdot \exp[\mathcal{O}(w \log D)]$, where the parameters w and D denote the treewidth and the diameter of the graph, respectively. More recently, a tour de force was achieved by Fomin et al. [53] who were the first to design parameterized algorithms with *polynomial dependency* on the treewidth, for Maximum Matching and Maximum Vertex-Flow. Furthermore they proved that for graphs with treewidth at most w, a tree decomposition of width $\mathcal{O}(w^2)$ can be computed in $\mathcal{O}(w^7 \cdot n \log n)$-time. We observe that their algorithm for Maximum Matching is *randomized*, whereas ours are deterministic.

The work of Fomin et al. has been continued by Iwata et al., who used a relative of treewidth called *tree-depth* as their parameter [75]. We are not aware of the study of another parameter than treewidth for polynomial-time solvable graph problems. However, some authors choose a different approach where they study the parameterization of a fixed graph problem for a broad range of graph invariants [11, 49, 84]. As an example, *clique-width* is part of the graph invariants used in the parameterized study of Triangle Listing [11]. Nonetheless, clique-width is not the main focus in [11]. Recently, Mertzios, Nichterlein and Niedermeier [84] proposed algorithms for Maximum Matching that run in time $\mathcal{O}(p^{\mathcal{O}(1)} \cdot (n+m))$, for several parameters p such as feedback vertex set or feedback edge set. Moreover, the authors in [84] suggest that Maximum Matching may become the "drosophila" of the study of the FPT algorithms in P. We advance in this research direction.

1.2 Our results

We study the parameterized complexity of several classical graph problems under a wide range of parameters such as clique-width and its upper bounds *modular-width* [35], *split-width* [92], *neighbourhood diversity* [81] and P_4*-sparseness* [8]. The results are summarized in Table 1.

Roughly, it turns out that some hardness assumptions for general graphs do not hold anymore for graph classes of bounded clique-width. This is the case in particular for Triangle Detection and other cycle problems that are subcubic equivalent to it such as, *e.g.*, Girth, that all can be solved in linear time, with quadratic dependency on the clique-width, with the help of dynamic programming (Theorems 2 and 3). The latter complements the results obtained for Triangle Listing in [11]. However we prove by an elementary argument that every P-FPT algorithm parameterized by clique-width can be transformed into a P-FPT algorithm parameterized by *treewidth*, with the same running-time up to polylogarithmic factors. The latter implies that the known hardness results for *distance problems* when using treewidth also hold when using clique-width (Theorems 6, 7 and 8). Unfortunately, these hardness results also hold for several other parameters related to clique-width (*i.e.*, see Fig. 7 for an illustration).

These negative results have motivated us to consider some upper bounds for clique-width as parameters, for which better results can be obtained than for clique-width. Another motivation stems from the fact that the existence of a parameterized (XP) algorithm for computing the clique-width of a graph remains a challenging open problem [25], whereas the other parameters considered in this work can be computed very efficiently. Indeed, we consider some upper bounds for clique-width that are defined via *linear-time* computable graph decompositions. Thus if these parameters are small enough, say, in $\mathcal{O}(n^{1-\varepsilon})$ for some $\varepsilon > 0$, we get truly subcubic or even truly subquadratic algorithms for a wide range of problems.

Problem	Parameterized complexity	SETH-based lower bounds
Triangle Detection, Triangle Counting, Girth	$\mathcal{O}(\mathsf{cw}(G)^2 \cdot (n+m))$	
Diameter, Eccentricities	$\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$ $\mathcal{O}(\mathsf{mw}(G)^2 \cdot n + m)$ $\mathcal{O}(\mathsf{mw}(G)^3 + n + m)$ $\mathcal{O}(\mathsf{q}(G)^3 + n + m)$	not in $2^{o(\mathsf{cw}(G))} \cdot n^{2-\varepsilon}$-time, for any $\varepsilon > 0$
Hyperbolicity	$\mathcal{O}(\mathsf{sw}(G)^3 \cdot n + m)$ $\mathcal{O}(\mathsf{mw}(G)^3 \cdot n + m)$ $\mathcal{O}(\mathsf{nd}(G)^4 + n + m)$ $\mathcal{O}(\mathsf{q}(G)^3 \cdot n + m)$	
Betweenness Centrality	$\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$ $\mathcal{O}(\mathsf{mw}(G)^2 \cdot n + m)$ $\mathcal{O}(\mathsf{nd}(G)^3 + n + m)$	
Maximum Matching	$\mathcal{O}(\mathsf{mw}(G)^4 \cdot n + m)$ $\mathcal{O}(\mathsf{q}(G)^4 \cdot n + m)$	

Table 1: Summary of positive results. We denote by $\mathsf{cw}(G)$ the clique-width of the graph G, $\mathsf{mw}(G)$ the modular-width, $\mathsf{sw}(G)$ the split-width, $\mathsf{nd}(G)$ the neighbourhood diversity, and $\mathsf{q}(G)$ the P_4-sparseness. Furthermore, all our positive results obtained for clique-width require a k-expression to be given as part of the input.

Graph parameters and decompositions considered

Let us describe the parameters considered in this work as follows. We provide first only an informal high level description. Formal definitions are postponed to Section 2.

Split Decomposition. A *join* is a set of edges inducing a complete bipartite subgraph. Roughly, clique-width can be seen as a measure of how easy it is to reconstruct a graph by adding joins between some vertex-subsets. A *split* is a join that is also an edge-cut. By using pairwise non crossing splits, termed "strong splits", we can decompose any graph into degenerate and prime subgraphs, that can be organized in a treelike manner. The latter is termed *split decomposition* [67].

We take advantage of the treelike structure of split decomposition in order to design dynamic programming algorithms for distance problems such as Diameter, Gromov Hyperbolicity and Betweenness Centrality (Theorems 9, 10 and 12, respectively). Although clique-width is also related to some treelike representations of graphs [33], the same cannot be done for clique-width as for split decomposition because the edges in the treelike representations for clique-width may not represent a join.

Modular Decomposition. Then, we can improve the results obtained with split decomposition by further restricting the type of splits considered. As an example, let (A, B) be a bipartition of the vertex-set that is obtained by removing a split. If every vertex of A is incident to some edges of the split then A is called a *module* of G. That is, for every vertex $v \in B$, v is either adjacent or nonadjacent to every vertex of A. The well-known *modular decomposition* of a graph is a hierarchical decomposition that partitions the vertices of the graph with respect to the modules [71]. Split decomposition is often presented as a refinement of modular decomposition [67]. We formalize the relationship between the two in Lemma 11, that allows us to also apply our methods for split decomposition to modular decomposition.

However, we can often do better with modular decomposition than with split decomposition. In particular, suppose we partition the vertex-set of a graph G into modules, and then we keep exactly one vertex per module. The resulting *quotient graph* G' keeps most of the distance properties of the input graph G. Therefore, in order to solve a distance problem for G, it is often the case that we only need to solve it for G'. We so believe that modular decomposition can be a powerful *Kernelization* tool in order to solve graph problems in P. As an application, we improve the running time for some of our algorithms, from time $\mathcal{O}(\mathsf{sw}(G)^{\mathcal{O}(1)} \cdot n + m)$ when parameterized by the *split-width* (maximum order of a prime subgraph in the split decomposition), to $\mathcal{O}(\mathsf{mw}(G)^{\mathcal{O}(1)} + n + m)$-time when parameterized by the *modular-width* (maximum order of a prime subgraph in the modular decomposition). See Theorem 14.

Furthermore, for some more graph problems, it may also be useful to further restrict the internal structures of modules. We briefly explore this possibility through a case study for *neighbourhood diversity*. Roughly, in this latter case we only consider modules that are either independent sets (false twins) or cliques (true twins). A new kernelization result is obtained for Hyperbolicity when parameterized by the neighbourhood diversity (Theorem 17). We can also reinterpret the main result from [89] as a kernelization algorithm for Betweenness Centrality when parameterized by the neighbourhood diversity (Theorem 18). It is worth pointing out that so far, we have been unable to obtain kernelization results for Hyperbolicity and Betweenness Centrality when only parameterized by the modular-width. It would be very interesting to prove separability results

between split-width, modular-width and neighbourhood diversity in the field of fully polynomial parameterized complexity.

Graphs with few P_4's. We finally use modular decomposition as our main tool for the design of new linear-time algorithms when restricted to graphs with few induced P_4's. The (q,t)-graphs have been introduced by Babel and Olariu in [7]. They are the graphs in which no set of at most q vertices can induce more than t paths of length four. Every graph is a (q,t)-graph for some large enough values of q and t. Furthermore when q and t are fixed constants, for $t \leq q-3$, the class of all the (q,t)-graphs has bounded clique-width [83]. We so define the P_4-sparseness of a graph G, denoted by $\mathsf{q}(G)$, as the minimum $q \geq 7$ such that G is a $(q, q-3)$-graph. The structure of the quotient graph of a $(q, q-3)$-graph, q being a constant, has been extensively studied and characterized in the literature [5, 7, 8, 6, 76]. We take advantage of these existing characterizations in order to generalize our algorithms with modular decomposition to $\mathcal{O}(\mathsf{q}(G)^{\mathcal{O}(1)} \cdot n + m)$-time algorithms (Theorems 19 and 21).

Let us give some intuition on how the P_4-sparseness can help in the design of improved algorithms for hard graph problems in P. We consider the class of *split graphs* (*i.e.*, graphs that can be bipartitioned into a clique and an independent set). Deciding whether a given split graph has diameter 2 or 3 is hard [18]. However, suppose now that the split graph is a $(q, q-3)$-graph G, for some fixed q. An induced P_4 in G has its two ends u, v in the independent set, and its two middle vertices are, respectively, in $N_G(u) \setminus N_G(v)$ and $N_G(v) \setminus N_G(u)$. Furthermore, when G is a $(q, q-3)$-graph, it follows from the characterization of [5, 7, 8, 6, 76] that either it has a quotient graph of bounded order $\mathcal{O}(q)$ or it is part of a well-structured subclass where the vertices of all neighbourhoods in the independent set follow a rather nice pattern (namely, spiders and a subclass of p-trees, see Section 2). As a result, the diameter of G can be computed in $\mathcal{O}(\max\{q^3, n+m\})$-time when G is a $(q, q-3)$ split graph. We generalize this result to every $(q, q-3)$-graph by using modular decomposition.

We want to further stress that the quotient graphs of $(q, q-3)$-graphs may have *unbounded* order. In particular, our results for these graphs are evidence that our general approach in the paper can also be applied to some classes of graphs with unbounded split-width or modular-width.

All the parameters considered in this work have already received some attention in the literature, especially in the design of FPT algorithms for NP-hard problems [6, 64, 67, 60, 92]. However, to the best of our knowledge, we are the first to study clique-width and its upper bounds for polynomial-time solvable problems. This distinction makes the analysis of our algorithms more delicate. Indeed, our approach in order to solve a polynomial-time solvable problem essentially consists in solving *weighted* versions of this problem for every component of the decomposition separately. In order to ensure that it gives us a P-FPT algorithm, we need to check whether the weighted version of our problem stays in P, and that the weight assignments can be computed very efficiently (say, in linear time). These above aspects can be overlooked in the design of "classical" FPT algorithms.

There do exist linear-time algorithms for Diameter, Maximum Matching and some other problems we study when restricted to some graph classes where the split-width or the P_4-sparseness is bounded (*e.g.*, cographs [101], distance-hereditary graphs [39, 40], P_4-tidy graphs [55], etc.). Nevertheless, we find the techniques used for these specific subclasses hardly generalize to the case where the graph has split-width or P_4-sparseness at most c, c being any fixed constant. For instance, the algorithm that is proposed in [40] for computing the diameter of a given distance-hereditary graph is based on some properties of LexBFS orderings. Distance-hereditary graphs

are exactly the graphs with split-width at most two [67]. However it does not look that simple to extend the properties found for their LexBFS orderings to bounded split-width graphs in general. As a byproduct of our approach, we also obtain new linear-time algorithms when restricted to well-known graph families such as cographs and distance-hereditary graphs.

Highlight of our Maximum Matching **algorithms**

Finally we emphasize our algorithms for Maximum Matching. Here we follow the suggestion of Mertzios, Nichterlein and Niedermeier [84] that Maximum Matching may become the "drosophila" of the study of the FPT algorithms in P. Precisely, we propose an $\mathcal{O}(p^4 \cdot n + m)$-time algorithm for Maximum Matching when parameterized either by modular-width or by the P_4-sparseness of the graph (Theorems 23 and 25). For every graph class where one of these two parameters is constantly bounded, this is linear time. We note that there already exist a few linear-time Maximum Matching algorithms for some graph classes where either the modular-width or the P_4-sparseness are constant [55, 101]. In fact, we unify and broadly generalize some of the techniques used in these papers. As a result, the combination of our Theorems 23 and 25 subsumes many algorithms that have been obtained for specific subclasses [55, 101].

Let us sketch the main lines of our approach. Our algorithms for Maximum Matching are recursive. Given a partition of the vertex-set into modules, first we compute a maximum matching for the subgraph induced by every module separately. Taking the union of all the outputted matchings gives a matching for the whole graph, but this matching is not necessarily maximum. So, we aim at increasing its cardinality by using augmenting paths [12]. In an unpublished paper [86], Novick followed a similar approach and, based on an integer programming formulation, he obtained an $\mathcal{O}(\mathsf{mw}(G)^{\mathcal{O}(\mathsf{mw}(G)^3)}n+m)$-time algorithm for Maximum Matching when parameterized by the modular-width. Our approach is more combinatorial than his.

Our contribution in this part is twofold. First we carefully study the possible ways an augmenting path can cross a module. Our analysis reveals that in order to compute a maximum matching in a graph we only need to consider augmenting paths of length $\mathcal{O}(\mathsf{mw}(G))$. Then, our second contribution is an efficient way to compute such paths. For that, we design a new type of characteristic graph of size $\mathcal{O}(\mathsf{mw}(G)^4)$. Similarly to the classical quotient graph that keeps most distance properties of the original graph, our new type of characteristic graph is tailored to enclose the main properties of the current matching in the graph. We believe that the design of new types of characteristic graphs can be a crucial tool in the design of improved algorithms for graph classes of bounded modular-width.

Recently, we discovered a distant relationship between our construction and the "substitutes" gadgets introduced by Gabow in his study of the Degree-Constrained Subgraph problem [58]. However, none of the reductions is implied by the other.

We have been able to extend our approach with modular decomposition to an $\mathcal{O}(q^4 \cdot n + m)$-time algorithm for computing a maximum matching in a given $(q, q-3)$-graph. However, a characterization of the quotient graph is not enough to do that. Specifically, we need to go deeper in the p-connectedness theory of [8] in order to better characterize the nontrivial modules in the graphs (Theorem 24). Furthermore our algorithm for $(q, q-3)$-graph not only makes use of the algorithm with modular decomposition. On our way to solve this case we have generalized different methods and reduction rules from the literature [78, 101], that is of independent interest.

Additional Related work. In a follow-up paper we obtained a Maximum Matching algorithm in quasi linear time for bounded split-width graphs [43]. For that we further used the aforementioned relationship between our construction in the current paper and the Degree-Constrained Subgraph problem (sometimes called b-Matching). Independently from our work in [43], Kratsch and Nelles proposed an improved version of our Maximum Matching algorithm for graphs of bounded modular-width which is also based on a reduction to b-Matching [80]. They also obtained several new linear-time algorithms for Maximum Flow and other polynomial-time solvable problems when parameterized by modular-width. Finally, in a second follow-up paper [44] we introduced several new reduction rules for Maximum Matching – based on vertex-orderings and modular decomposition. Doing so, we generalized our algorithms in this current paper to several new classes of graphs where the quotient graphs may be unbounded.

1.3 Organization of the paper

In Section 2 we introduce definitions and basic notations.

Then, in Section 3 we show FPT algorithms when parameterized by the clique-width. The problems considered are Triangle Counting and Girth. To the best of our knowledge, we present the first known polynomial parameterized algorithm for Girth (Theorem 3). Roughly, the main idea behind our algorithms is that given a labeled graph G obtained from a k-expression, we can compute a minimum-length cycle for G by keeping up to date the pairwise distances between every two label classes. Hence, if a k-expression of length L is given as part of the input we obtain algorithms running in time $\mathcal{O}(k^2L)$ and space $\mathcal{O}(k^2)$.

In Section 4 we consider distance related problems, namely: Diameter, Eccentricities, Hyperbolicity and Betweenness Centrality.

We start proving, in Section 4.2, that none of the above problems can be solved in time $2^{o(k)}n^{2-\varepsilon}$, for any $\varepsilon > 0$, when parameterized by the clique-width (Theorems 6—8). These are the first known hardness results for clique-width in the field of "Hardness in P". Furthermore, as it is often the case in this field, our results are conditioned on the Strong Exponential Time Hypothesis [74]. In summary, we take advantage of recent hardness results obtained for *bounded-degree* graphs [47]. Clique-width and treewidth can only differ by a constant-factor in the class of bounded-degree graphs [31, 70]. Therefore, by combining the hardness constructions for bounded-treewidth graphs and for bounded-degree graphs, we manage to derive hardness results for graph classes of bounded clique-width. We also show how to derive a weaker, but more general hardness result for clique-width by using a classical relationship between clique-width and pathwidth (*e.g.*, see [48]).

In Section 4.3 we describe fully polynomial FPT algorithms for Diameter, Eccentricity, Hyperbolicity and Betweenness centrality parameterized by the split-width. Our algorithms use split-decomposition as an efficient preprocessing method. Roughly, we define weighted versions for every problem considered (some of them admittedly technical). In every case, we prove that solving the original distance problem can be reduced in linear time to the solving of its weighted version for every subgraph of the split decomposition separately.

Then, in Section 4.4 we apply the results from Section 4.3 to modular-width. First, since for any graph G we have $\mathsf{sw}(G) \leq \mathsf{mw}(G) + 1$, all our algorithms parameterized by split-width are also algorithms parameterized by modular-width. Moreover for Eccentricities, and for Hyperbolicity and Betweenness Centrality when parameterized by the neighbourhood diversity, we

show that it is sufficient to only process the quotient graph of G. We thus obtain algorithms that either run in $\mathcal{O}(\mathsf{mw}(G)^{\mathcal{O}(1)} + n + m)$-time, or $\mathcal{O}(\mathsf{nd}(G)^{\mathcal{O}(1)} + n + m)$-time, for all these problems.

In Section 4.5 we generalize our previous algorithms to be applied to the $(q, q-3)$-graphs. We obtain our results by carefully analyzing the cases where the quotient graph has size $\Omega(q)$. These cases are given by Lemma 4.

Section 5 is dedicated to our main result, linear-time algorithms for Maximum Matching. First in Section 5.1 we propose an algorithm that is parameterized by the modular-width and runs in time $\mathcal{O}(\mathsf{mw}(G)^4 \cdot n + m)$. In Section 5.2 we generalize this algorithm to $(q, q-3)$-graphs.

Finally, in Section 6 we discuss applications to other graph classes with unbounded clique-width. Results of this paper were partially presented at the SODA'18 conference [29].

2 Preliminaries

We use standard graph terminology from [16, 37]. Graphs in this study are finite, simple (hence without loops or multiple edges) and unweighted — unless stated otherwise. Furthermore we make the standard assumption that graphs are encoded as adjacency lists.

We start introducing a few basic notations. Given a graph $G = (V, E)$, let $n = |V|$ be its order (number of vertices) and $m = |E|$ be its size (number of edges). For every vertex-subset $S \subseteq V$ we denote by $G[S] = (S, E \cap (S \times S))$ the subgraph of G that is induced by S. For every $v \in V$, we denote its open neighbourhood by $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$. Similarly, the closed neighbourhood of any vertex v is denoted by $N_G[v] = N_G(v) \cup \{v\}$. We will remove the subscript if the graph G is clear from the context. The characteristic function of S, denoted by $[v \in S]$ is equal to 1 if $v \in S$ and to 0 otherwise. Finally, we will denote by $[1; n] = \{1, 2, \ldots, n\}$ the set of the n first positive integers.

We want to prove the existence, or the nonexistence, of graph algorithms with running time of the form $k^{\mathcal{O}(1)} \cdot (n + m)$, k being some fixed graph parameter. In what follows, we introduce the graph parameters considered in this work.

Clique-width

A labeled graph is given by a pair $\langle G, \ell \rangle$ where $G = (V, E)$ is a graph and $\ell : V \to \mathbb{N}$ is called a labeling function. A *k-expression* can be seen as a sequence of operations for constructing a labeled graph $\langle G, \ell \rangle$, where the allowed four operations are:

1. Addition of a new vertex v with label i (the labels are taken in $\{1, 2, \ldots, k\}$), denoted $i(v)$;
2. Disjoint union of two labeled graphs $\langle G_1, \ell_1 \rangle$ and $\langle G_2, \ell_2 \rangle$, denoted $\langle G_1, \ell_1 \rangle \oplus \langle G_2, \ell_2 \rangle$;
3. Addition of a join between the set of vertices labeled by i and the set of vertices labeled by j, where $i \neq j$, denoted $\eta(i, j)$;
4. Renaming label i to label j, denoted $\rho(i, j)$.

See Fig. 1 for an example of these operations. The *clique-width* of G, denoted by $\mathsf{cw}(G)$, is the minimum k such that, for some labeling ℓ, the labeled graph $\langle G, \ell \rangle$ admits a k-expression [32]. We refer to [34] and the references cited therein for a survey of the many applications of clique-width in the field of parameterized complexity.

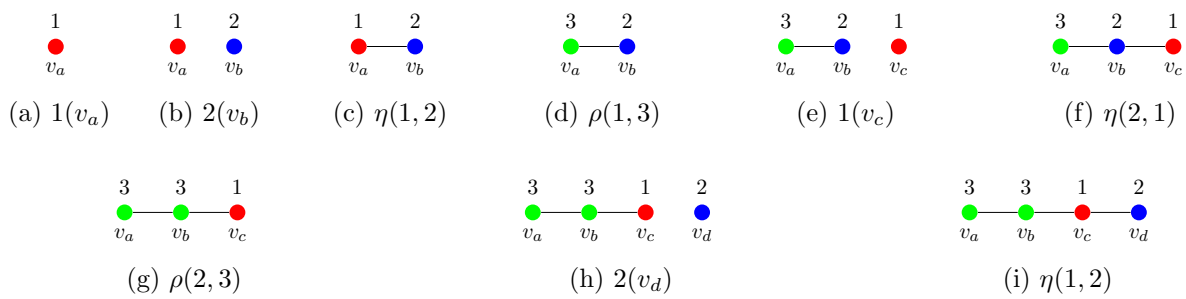

Figure 1: A 3-expression for the path P_4.

Computing the clique-width of a given graph is NP-hard [48]. However, on a more positive side the graphs with clique-width two are exactly the cographs and they can be recognized in linear time [26, 35]. Clique-width three graphs can also be recognized in polynomial time [25]. The parameterized complexity of computing the clique-width is an open question. In what follows, we focus on upper bounds on clique-width that are derived from some graph decompositions.

Modular-width

A *module* in a graph $G = (V, E)$ is any subset $M \subseteq V$ such that for any $v \in V \setminus M$, either $M \subseteq N_G(v)$ or $M \cap N_G(v) = \emptyset$. Note that $\emptyset$, V, and $\{v\}$ for every $v \in V$ are trivial modules of G. A graph is called *prime* for modular decomposition if it only has trivial modules.

A module M is *strong* if it does not overlap any other module, *i.e.*, for any module M' of G, either one of M or M' is contained in the other or M and M' do not intersect. Furthermore, let us denote by $\mathcal{M}(G)$ the family of all inclusion wise maximal strong modules of G that do not contain all the vertices of G (*i.e.*, they are proper subsets of V). The *quotient graph* of G is the graph G' with vertex-set $\mathcal{M}(G)$ and an edge between every two $M, M' \in \mathcal{M}(G)$ such that every vertex of M is adjacent to every vertex of M'.

Modular decomposition is based on the following structure theorem from Gallai.

Theorem 1 ([62]). *For an arbitrary graph G exactly one of the following conditions is satisfied.*

1. *G is disconnected;*
2. *its complement $\overline{G}$ is disconnected;*
3. *or its quotient graph G' is prime for modular decomposition.*

Theorem 1 suggests the following recursive procedure in order to decompose a graph, that is sometimes called modular decomposition. If $G = G'$ (*i.e.*, G is complete, edgeless or prime for modular decomposition) then we output G. Otherwise, we output the quotient graph G' of G and, for every strong module M of G, the modular decomposition of $G[M]$. The modular decomposition of a given graph G can be computed in linear time [98]. More precisely, we can compute in linear time a rooted labeled tree whose nodes are the subgraphs of the modular decomposition; each node is either labeled prime, complete or edgeless. See Fig. 2 for an example.

Furthermore, by Theorem 1 the subgraphs from the modular decomposition are either edgeless, complete, or prime for modular decomposition. The *modular-width* of G, denoted by $\mathsf{mw}(G)$, is the

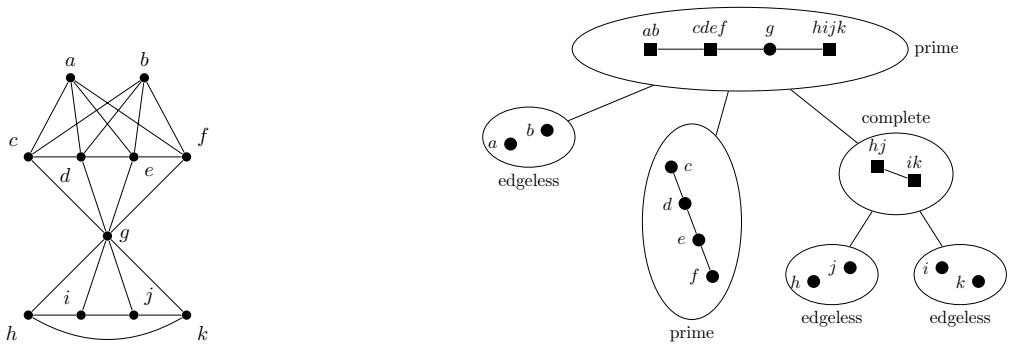

Figure 2: A graph and its modular decomposition.

minimum $k \geq 2$ such that any prime subgraph in the modular decomposition has order (number of vertices) at most k [1]. The relationship between clique-width and modular-width is as follows.

Lemma 1 ([34]). *For every graph G, we have $\mathsf{cw}(G) \leq \mathsf{mw}(G)$, and we can construct a $\mathsf{mw}(G)$-expression defining G in linear time.*

We refer to [71] for a survey on modular decomposition. In particular, graphs with modular-width two are exactly the cographs, that follows from the existence of a cotree [97]. Cographs enjoy many algorithmic properties, including a linear-time algorithm for Maximum Matching [101]. Furthermore, in [60] Gajarskỳ, Lampis and Ordyniak prove that for some $W[1]$-hard problems when parameterized by clique-width there exist FPT algorithms when parameterized by modular-width.

Split-width

A *split* (A, B) in a *connected* graph $G = (V, E)$ is a partition $V = A \cup B$ such that: $\min\{|A|, |B|\} \geq 2$; and there is a complete join between the vertices of $N_G(A)$ and $N_G(B)$. For every split (A, B) of G, let $a \in N_G(B)$, $b \in N_G(A)$ be arbitrary. The vertices a, b are termed *split marker vertices*. We can compute a "simple decomposition" of G into the two subgraphs $G_A = G[A \cup \{b\}]$ and $G_B = G[B \cup \{a\}]$.

There are two cases of "indecomposable" graphs. Degenerate graphs are such that every bi-partition of their vertex-set is a split. They are exactly the complete graphs and the stars [36]. A graph is prime for split decomposition if it has no split.

A split decomposition of a connected graph G is obtained by applying recursively a simple decomposition, until all the subgraphs obtained are either degenerate or prime. A split decomposition of an *arbitrary* graph G is the union of a split decomposition for each of its connected components. Every graph has a canonical split decomposition, with minimum number of subgraphs, that can be computed in linear time [22]. The *split-width* of G, denoted by $\mathsf{sw}(G)$, is the minimum $k \geq 2$ such that any prime subgraph in the canonical split decomposition of G has order at most k. See Fig. 3 for an illustration.

Lemma 2 ([92]). *For every graph G, we have $\mathsf{cw}(G) \leq 2 \cdot \mathsf{sw}(G) + 1$, and a $(2 \cdot \mathsf{sw}(G) + 1)$-expression defining G can be constructed in linear time.*

[1] This term has another meaning in [91]. We rather follow the terminology from [35].

We refer to [64, 67, 92] for some algorithmic applications of split decomposition. In particular, graphs with split-width at most two are exactly the distance-hereditary graphs [9]. Linear-time algorithms for solving Diameter and Maximum Matching for (subclasses of) distance-hereditary graphs are presented in [40, 39].

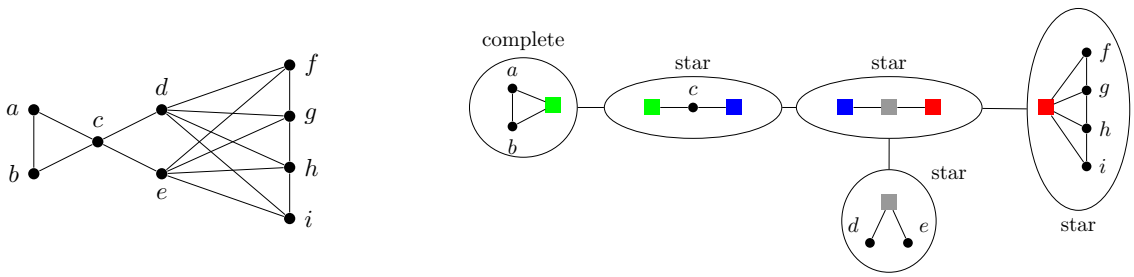

Figure 3: A graph and its split decomposition.

We stress that split decomposition can be seen as a refinement of modular decomposition. Indeed, if M is a module of $G = (V, E)$ and $\min\{|M|, |V \setminus M|\} \geq 2$ then $(M, V \setminus M)$ is a split. In what follows, we prove most of our results with the more general split decomposition.

Graphs with few P_4's

A (q, t)-graph $G = (V, E)$ is such that for any $S \subseteq V$, $|S| \leq q$, S induces at most t paths on four vertices [7]. The *P_4-sparseness* of G, denoted by $\mathsf{q}(G)$, is the minimum $q \geq 7$ such that G is a $(q, q-3)$-graph.

Lemma 3 ([83])**.** *For every $q \geq 7$, every $(q, q-3)$-graph G has clique-width at most q, and we can compute a q-expression defining G in linear time.*

The algorithmic properties of several subclasses of $(q, q-3)$-graphs have been considered in the literature. We refer to [8] for a survey. Furthermore, there exists a canonical decomposition of $(q, q-3)$-graphs, sometimes called the *primeval decomposition*, that can be computed in linear time [10]. Primeval decomposition can be seen as an intermediate between modular and split decomposition. We postpone the presentation of primeval decomposition until Section 5. Until then, we state the results in terms of modular decomposition.

More precisely, given a $(q, q-3)$-graph G, the prime subgraphs in its modular decomposition may be of super-constant size $\Omega(q)$. However, if they are then they are part of one of the well-structured graph classes that we detail next.

A *disc* is either a cycle C_n, or a co-cycle $\overline{C_n}$, for some $n \geq 5$.

A *spider* $G = (S \cup K \cup R, E)$ is a graph with vertex set $V = S \cup K \cup R$ and edge set E such that:

1. (S, K, R) is a partition of V and R may be empty;
2. the subgraph $G[K \cup R]$ induced by K and R is the complete join $K \oplus R$, and K separates S and R, i.e. any path between a vertex in S and a vertex in R contains a vertex in K;
3. S is a stable set, K is a clique, $|S| = |K| \geq 2$, and there exists a bijection $f : S \longrightarrow K$ such that, either for all vertices $s \in S$, $N(s) \cap K = K - \{f(s)\}$ or $N(s) \cap K = \{f(s)\}$. Roughly

(a) Thin spider. (b) Thick spider.

Figure 4: Spiders.

speaking, the edges between S and K are either a matching or an anti-matching. In the former case or if $|S| = |K| \leq 2$, G is called *thin*, otherwise G is *thick*. See Fig. 4.

If furthermore $|R| \leq 1$ then we call G a *prime spider*.

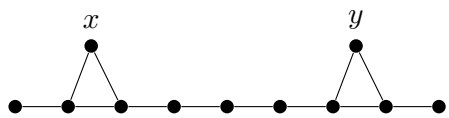

Figure 5: Spiked p-chain P_k.

Let $P_k = (v_1, v_2, v_3, \ldots, v_k),\ k \geq 6$ be a path of length at least five. A *spiked p-chain* P_k is a supergraph of P_k, in which we possibly added any of the two vertices x, y such that: $N(x) = \{v_2, v_3\}$ and $N(y) = \{v_{k-2}, v_{k-1}\}$, respectively. See Fig. 5. Note that one or both of x and y may be missing. In particular, P_k is a spiked p-chain P_k. Spiked p-chains P_k are $(7, 4)$-graphs, and so are their complements. We call *spiked p-chain* $\overline{P_k}$ the complement of a spiked p-chain P_k.

Let Q_k be the graph with vertex-set $\{v_1, v_2, \ldots, v_k\},\ k \geq 6$ such that: $N_{Q_k}(v_{2i-1}) = \{v_{2j} \mid j \leq i,\ j \neq i-1\}$ and $N_{Q_k}(v_{2i}) = \{v_{2j} \mid j \neq i\} \cup \{v_{2j-1} \mid j \geq i,\ j \neq i+1\}$, for every $i \geq 1$. A *spiked p-chain* Q_k is a supergraph of Q_k, possibly with the additional vertices $z_2, z_3, \ldots, z_{k-5}$ such that:

- $N(z_{2i-1}) = \{v_{2j} \mid j \in [1; i]\} \cup \{z_{2j} \mid j \in [1; i-1]\}$;

- $\overline{N(z_{2i})} = \{v_{2j-1} \mid j \in [1; i+1]\} \cup \{z_{2j-1} \mid j \in [2; i]\}$

Any of the vertices z_i can be missing, so, in particular, Q_k is a spiked p-chain Q_k. See Fig. 6. A *spiked p-chain* $\overline{Q_k}$ is the complement of a spiked p-chain Q_k.

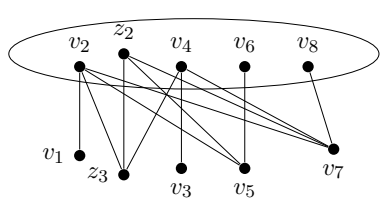

Figure 6: Spiked p-chain Q_k.

Finally, we say that a graph is a *prime p-tree* if it is either: a spiked p-chain P_k, a spiked p-chain $\overline{P_k}$, a spiked p-chain Q_k, a spiked p-chain $\overline{Q_k}$, or part of the seven graphs of order at most 7 that are listed in [83].

Lemma 4 ([6, 83]). *Let G be a connected $(q, q-3)$-graph, for some $q \geq 7$, such that G and $\overline{G}$ are connected. Then, one of the following must hold for its quotient graph G':*

- *either G' is a prime spider;*
- *or G' is a disc;*
- *or G' is a prime p-tree;*
- *or $|V(G')| \leq q$.*

A simpler version of Lemma 4 holds for the subclass of $(q, q-4)$-graphs:

Lemma 5 ([6]). *Let G be a connected $(q, q-4)$-graph, for some $q \geq 4$, such that G and $\overline{G}$ are connected. Then, one of the following must hold for its quotient graph G':*

- *G' is a prime spider;*
- *or $|V(G')| \leq q$.*

The subclass of $(q, q-4)$-graphs has received more attention in the literature than $(q, q-3)$-graphs. Our results hold for the more general case of $(q, q-3)$-graphs.

3 Cycle problems on bounded clique-width graphs

Clique-width is the smallest parameter that is considered in this work. We start studying the possibility for $k^{\mathcal{O}(1)} \cdot (n+m)$-time algorithms on graphs with clique-width at most k. Positive results are obtained for two variations of Triangle Detection, namely Triangle Counting and Girth. We define the problems studied in Section 3.1, then we describe the algorithms in order to solve these problems in Section 3.2.

3.1 Problems considered

We start introducing our basic cycle problem.

Problem 1 (Triangle Detection).

Input: *A graph G.*

Question: *Does there exist a triangle in G?*

Note that for general graphs, Triangle Detection is conjectured not to be solvable in $\mathcal{O}(n^{3-\varepsilon})$-time, for any $\varepsilon > 0$, with a combinatorial algorithm (*i.e.*, not using algebraic methods) [100]. It is also conjectured not to be solvable in $\mathcal{O}(n^{\omega-\varepsilon})$-time for any $\varepsilon > 0$, with ω being the exponent for fast matrix multiplication [2]. Our results in this section show that such assumptions do not hold when restricted to bounded clique-width graphs.

More precisely, we next describe fully polynomial parameterized algorithms for the two following generalizations of Triangle Detection.

Problem 2 (Triangle Counting).

Input: *A graph G.*

Output: *The number of triangles in G.*

Problem 3 (Girth).

Input: *A graph G.*

Output: *The girth of G, that is the minimum size of a cycle in G.*

In [100], the three of Triangle Detection, Triangle Counting and Girth are proved to be subcubic equivalent when restricted to combinatorial algorithms.

3.2 Algorithms

Roughly, our algorithms in what follows are based on the following observation. Given a labeled graph $\langle G, \ell \rangle$ (obtained from a k-expression), in order to detect a triangle in G, resp. a minimum-length cycle in G, we only need to store the adjacencies, resp. the distances, between every two label classes. Hence, if a k-expression of length L is given as part of the input we obtain algorithms running in time $\mathcal{O}(k^2 L)$ and space $\mathcal{O}(k^2)$.

Our first result is for Triangle Counting (Theorem 2). Specifically, we propose an algorithm whose main task is to count the number of new triangles that appeared in the graph after a join operation occured. We recall that a join induces a complete *bipartite* subgraph, that we denote by B. It implies that any triangle can have at most two edges that are contained into the join. Therefore we essentially need to keep track, in the subgraph $G \setminus E(B)$ before the join operation, of the number of edges with their two ends in the same side of B *and* of the number of paths of length two with an end in each side of B. – Indeed, these two cases correspond to the situations where a triangle has two edges or one edge in the join, respectively. –

Our algorithm shares some similarities with a recent $\mathcal{O}(n^2 + \mathsf{cw}(G)^2 \cdot n + t)$-time algorithm for listing all t triangles in a graph [11]. However, unlike the authors in [11], we needn't use the notion of k-modules in our algorithms. Furthermore, since we only ask for *counting* triangles, and not to list them, we obtain a better time complexity than in [11]. To better understand why this is so, let us consider the case where a join is added in a labeled graph $\langle G, \ell \rangle$ between the set V_i of vertices labeled by i and the set V_j of vertices labeled by j. For every induced path of length two with one end in V_i and the other end in V_j, we create one triangle. Then, if we just want to *count* the number of new triangles which are created that way, it suffices to store the number of induced paths of length two with their ends in V_i and V_j, respectively. Overall, we so partition the induced paths of length two in G into $\mathcal{O}(k^2)$ equivalence classes; we will show in our proof that the cardinality of all these classes can also be updated in $\mathcal{O}(k^2)$-time after any operation occured in a corresponding k-expression of the graph. Unfortunately if instead of counting we wish to *list* the triangles in G, then it seems that more information needs to be stored as we will sometimes need to *enumerate* all the paths of length two within the same equivalence class.

Theorem 2. *For every $G = (V, E)$,* Triangle Counting *can be solved in $\mathcal{O}(k^2 \cdot (n + m))$-time if a k-expression of G is given.*

Proof. We need to assume the k-expression is *irredundant*, that is, when we add a complete join between the vertices labeled i and the verticed labeled j, there was no edge before between these two subsets. Given a k-expression of G, an irredundant k-expression can be computed in linear time [35]. Then, we proceed by dynamic programming on the irredundant k-expression.

More precisely, let $\langle G, \ell \rangle$ be a labeled graph, $\ell : V \to \{1, \dots, k\}$. We denote by $T(\langle G, \ell \rangle)$ the number of triangles in G. In particular, $T(\langle G, \ell \rangle) = 0$ if G is empty. Furthermore, $T(\langle G, \ell \rangle) = T(\langle G', \ell' \rangle)$ if $\langle G, \ell \rangle$ is obtained from $\langle G', \ell' \rangle$ by: the addition of a new vertex with any label, or the identification of two labels. If $\langle G, \ell \rangle$ is the disjoint union of $\langle G_1, \ell_1 \rangle$ and $\langle G_2, \ell_2 \rangle$ then $T(\langle G, \ell \rangle) = T(\langle G_1, \ell_1 \rangle) + T(\langle G_2, \ell_2 \rangle)$.

Finally, suppose that $\langle G, \ell \rangle$ is obtained from $\langle G', \ell' \rangle$ by adding a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. For every $p, q \in \{1, \dots, k\}$, we denote by $m_{p,q}$ the number of edges in $\langle G', \ell' \rangle$ with one end in V_p and the other end in V_q. Let $n_{p,q}$ be the number of (non necessarily induced) P_3's with an end in V_p and the other end in V_q. Note that we are only interested in the number of *induced* P_3's for our algorithm, but this looks more challenging to compute. Nevertheless, since the k-expression is irredundant, $n_{i,j}$ is exactly the number of induced P_3's with one end in V_i and the other in V_j, and the middle vertices on such paths cannot be in $V_i \cup V_j$. Furthermore after the join is added we get: $|V_i|$ new triangles per edge in $G'[V_j]$, $|V_j|$ new triangles per edge in $G'[V_i]$, and one triangle for every P_3 with one end in V_i and the other in V_j. Summarizing:

$$T(\langle G, \ell \rangle) = T(\langle G', \ell' \rangle) + |V_j| \cdot m_{i,i} + |V_i| \cdot m_{j,j} + n_{i,j}.$$

In order to derive the claimed time bound, we are now left to prove that, after any operation, we can update the values $m_{p,q}$ and $n_{p,q}$, $p, q \in \{1, \dots, k\}$, in $\mathcal{O}(k^2)$-time. Clearly, these values cannot change when we add a new (isolated) vertex, with any label, and they can be updated by simple summation when we take the disjoint union of two labeled graphs. We now need to distinguish between the two remaining cases. In what follows, let $m'_{p,q}$ and $n'_{p,q}$ represent the former values.

- Suppose that label i is identified with label j. Then:

$$m_{p,q} = \begin{cases} 0 & \text{if } i \in \{p, q\} \\ m'_{i,i} + m'_{i,j} + m'_{j,j} & \text{if } p = q = j \\ m'_{p,j} + m'_{p,i} & \text{if } q = j,\ p \notin \{i, j\} \\ m'_{j,q} + m'_{i,q} & \text{if } p = j,\ q \notin \{i, j\} \\ m'_{p,q} & \text{else} \end{cases},$$

$$n_{p,q} = \begin{cases} 0 & \text{if } i \in \{p, q\} \\ n'_{j,j} + n'_{j,i} + n'_{i,i} & \text{if } p = q = j \\ n'_{p,j} + n'_{p,i} & \text{if } q = j,\ p \notin \{i, j\} \\ n'_{j,q} + n'_{i,q} & \text{if } p = j,\ q \notin \{i, j\} \\ n'_{p,q} & \text{else} \end{cases}.$$

- Otherwise, suppose that we add a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. Then, since the k-expression is irredundant:

$$m_{p,q} = \begin{cases} |V_i| \cdot |V_j| & \text{if } \{i, j\} = \{p, q\} \\ m'_{p,q} & \text{else} \end{cases}.$$

For every $u_i, v_i \in V_i$ and $w_j \in V_j$ we create a new P_3 (u_i, w_j, v_i). Similarly, for every $u_j, v_j \in V_j$ and $w_i \in V_i$ we create a new P_3 (u_j, w_i, v_j). These are the only new P_3's with two edges from the complete join. Furthermore, for every edge $\{u_i, v_i\}$ in V_i and for every $w_j \in V_j$ we can create the two new P_3's (u_i, v_i, w_j) and (v_i, u_i, w_j). Similarly, for every edge $\{u_j, v_j\}$ in V_j and for every $w_i \in V_i$ we can create the two new P_3's (u_j, v_j, w_i) and (v_j, u_j, w_i). Finally, for every edge $\{v, u_j\}$ with $u_j \in V_j,\ v \notin V_i \cup V_j$, we create $|V_i|$ new P_3's, and for every edge $\{v, u_i\}$ with $u_i \in V_i,\ v \notin V_i \cup V_j$, we create $|V_j|$ new P_3's. Altogether combined, we deduce the following update rules:

$$n_{p,q} = \begin{cases} n'_{i,i} + |V_i| \cdot |V_j| \cdot (|V_i| - 1)/2 & \text{if } p = q = i \\ n'_{j,j} + |V_j| \cdot |V_i| \cdot (|V_j| - 1)/2 & \text{if } p = q = j \\ n'_{i,j} + 2 \cdot |V_j| \cdot m'_{i,i} + 2 \cdot |V_i| \cdot m'_{j,j} & \text{if } \{p, q\} = \{i, j\} \\ n'_{i,q} + |V_i| \cdot m'_{j,q} & \text{if } p = i, q \notin \{i, j\} \\ n'_{p,i} + |V_i| \cdot m'_{p,j} & \text{if } q = i, p \notin \{i, j\} \\ n'_{j,q} + |V_j| \cdot m'_{i,q} & \text{if } p = j, q \notin \{i, j\} \\ n'_{p,j} + |V_j| \cdot m'_{p,i} & \text{if } q = j, p \notin \{i, j\} \\ n'_{p,q} & \text{else} \end{cases}.$$

□

Our next result is about computing the *girth* of a graph (size of a smallest cycle). To the best of our knowledge, the following Theorem 3 gives the first known polynomial parameterized algorithm for Girth.

Theorem 3. *For every $G = (V, E)$,* Girth *can be solved in $\mathcal{O}(k^2 \cdot (n + m))$-time if a k-expression of G is given.*

Proof. The same as for Theorem 2, we assume the k-expression to be irredundant. It can be enforced up to linear-time preprocessing [35]. We proceed by dynamic programming on the k-expression. More precisely, let $\langle G, \ell \rangle$ be a labeled graph, $\ell : V \to \{1, \ldots, k\}$. We denote by $\mu(\langle G, \ell \rangle)$ the girth of G. By convention, $\mu(\langle G, \ell \rangle) = +\infty$ if G is empty, or more generally if G is a forest. Furthermore, $\mu(\langle G, \ell \rangle) = \mu(\langle G', \ell' \rangle)$ if $\langle G, \ell \rangle$ is obtained from $\langle G', \ell' \rangle$ by: the addition of a new vertex with any label, or the identification of two labels. If $\langle G, \ell \rangle$ is the disjoint union of $\langle G_1, \ell_1 \rangle$ and $\langle G_2, \ell_2 \rangle$ then $\mu(\langle G, \ell \rangle) = \min\{\mu(\langle G_1, \ell_1 \rangle), \mu(\langle G_2, \ell_2 \rangle)\}$.

Suppose that $\langle G, \ell \rangle$ is obtained from $\langle G', \ell' \rangle$ by adding a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. For every $p, q \in \{1, \ldots, k\}$, we define $d_{p,q}$ as the minimum length of a *nonempty* path with an end in V_p and an end in V_q. The requirement for the path being nonempty, *i.e.*, with two different ends, is important only if $p = q$. Furthermore, note that such a path as defined above may not exist. So, we may have $d_{p,q} = +\infty$. Then, let us consider a minimum-size cycle C of G. We distinguish between four cases.

- If C does not contain an edge of the join, then it is a cycle of G'.
- Else, suppose that C contains exactly one edge of the join. Then removing this edge leaves a V_iV_j-path in G'; this path has length at least $d_{i,j}$. Conversely, if $d_{i,j} \neq +\infty$ then there exists a cycle of length $1 + d_{i,j}$ in G, and so, $\mu(\langle G, \ell \rangle) \leq 1 + d_{i,j}$.

- Else, suppose that C contains exactly two edges of the join. In particular, since C is of minimum-size, and so, it is an induced cycle, the two edges of the join in C must have a common end in the cycle. It implies that removing the two edges from C leaves a path of G' with either its two ends in V_i or its two ends in V_j. Such paths have respective length at least $d_{i,i}$ and $d_{j,j}$. Conversely, there exist closed walks of respective length $2 + d_{i,i}$ and $2 + d_{j,j}$ in G. Hence, $\mu(\langle G, \ell\rangle) \leq 2 + \min\{d_{i,i}, d_{j,j}\}$.

- Otherwise, C contains at least three edges of the join. Since C is induced, it implies that C is a cycle of length four with two vertices in V_i and two vertices in V_j. Such a (non necessarily induced) cycle exists if and only if $\min\{|V_i|, |V_j|\} \geq 2$.

Summarizing:

$$\mu(\langle G, \ell\rangle) = \begin{cases} \min\{\mu(\langle G', \ell'\rangle),\ 1 + d_{i,j},\ 2 + d_{i,i}, 2 + d_{j,j}\} & \text{if } \min\{|V_i|, |V_j|\} = 1 \\ \min\{\mu(\langle G', \ell'\rangle),\ 1 + d_{i,j},\ 2 + d_{i,i}, 2 + d_{j,j},\ 4\} & \text{otherwise.} \end{cases}$$

In order to derive the claimed time bound, we are now left to prove that, after any operation, we can update the values $d_{p,q},\ p, q \in \{1, \ldots, k\}$, in $\mathcal{O}(k^2)$-time. As we shall see next, the update rules for that are rather standard, except for the particular case when $p = q$. Indeed, to understand the difficulty in this latter case, suppose that we add a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. Then, the distance $d_{p,p}$ might be decreased by using two consecutive edges of the join, *e.g.*, we follow a V_pV_i-path, we pass by some vertex in V_j and then we follow *another* V_iV_p-path. In order to make sure that such shortcut is valid, we need to keep track of at least two V_pV_i-paths (if any two such paths exist).

Motivated by this above observation, we store for any pair p, q of *distinct* labels a pair $\langle u_{p,q}, d_{p,q}\rangle$ such that: $u_{p,q} \in V_p$, and $dist(u_{p,q}, V_q) = dist(V_p, V_q) = d_{p,q}$. Observe that $d_{p,q} = d_{q,p}$; however we have $u_{p,q} \neq u_{q,p}$, and possibly $dist(u_{p,q}, u_{q,p}) > d_{p,q}$. Furthermore, unless V_p is a singleton, we also store $\langle v_{p,q}, \ell_{p,q}\rangle$ such that: $v_{p,q} \in V_p \setminus u_{p,q}$, and $dist(v_{p,q}, V_q) = dist(V_p \setminus u_{p,q}, V_q) = \ell_{p,q}$ (second closest vertex in V_p to V_q).

Clearly, these values cannot change when we add a new (isolated) vertex v, with any label i, unless $V_i \setminus v$ is either empty or a singleton. In the former case, for any $p \neq i$ we can initialize $d_{i,p} = d_{p,i} = \ell_{p,i} = +\infty$; furthermore, we set $u_{i,p} = v$ and we pick $u_{p,i}, v_{p,i} \in V_p$ arbitrarily. In the latter case, we can set $v_{i,p} = v$ and $\ell_{i,p} = +\infty$. No further changes are necessary.

Another simple case is the disjoint union $\langle G_1, \ell_1\rangle \oplus \langle G_2, \ell_2\rangle$ of two labeled graphs. Indeed, for every $p \neq q$, we consider the at most four pairs $\langle u^1_{p,q}, d^1_{p,q}\rangle$, $\langle v^1_{p,q}, \ell^1_{p,q}\rangle$ in G_1 and $\langle u^2_{p,q}, d^2_{p,q}\rangle$, $\langle v^2_{p,q}, \ell^2_{p,q}\rangle$ in G_2. We set $\langle u_{p,q}, d_{p,q}\rangle = \langle u^1_{p,q}, d^1_{p,q}\rangle$ if $d^1_{p,q} \leq d^2_{p,q}$, otherwise we set $\langle u_{p,q}, d_{p,q}\rangle = \langle u^2_{p,q}, d^2_{p,q}\rangle$. Without loss of generality, $d^1_{p,q} \leq d^2_{p,q}$. Then, we set $\langle v_{p,q}, \ell_{p,q}\rangle = \langle u^2_{p,q}, d^2_{p,q}\rangle$ if $d^2_{p,q} \leq \ell^1_{p,q}$, otherwise we set $\langle v_{p,q}, \ell_{p,q}\rangle = \langle v^1_{p,q}, \ell^1_{p,q}\rangle$.

We now need to distinguish between the two remaining cases. In what follows, let $\langle u'_{p,q}, d'_{p,q}\rangle$ and $\langle v'_{p,q}, \ell'_{p,q}\rangle$ represent the former values.

- Suppose that label i is identified with label j. Let $q \notin \{i, j\}$ be any other label.

  We consider the at most four pairs $\langle u'_{i,q}, d'_{i,q}\rangle$, $\langle u'_{j,q}, d'_{j,q}\rangle$ and $\langle v'_{i,q}, \ell'_{i,q}\rangle$, $\langle v'_{j,q}, \ell'_{j,q}\rangle$. W.l.o.g. $d'_{i,q} \leq d'_{j,q}$, and so, we set $\langle u_{j,q}, d_{j,q}\rangle = \langle u'_{i,q}, d'_{i,q}\rangle$. Furthermore, $\langle v_{j,q}, \ell_{j,q}\rangle$ can be taken as one of $\langle u'_{j,q}, d'_{j,q}\rangle$ (if $d'_{j,q} \leq \ell'_{i,q}$) or $\langle v'_{i,q}, \ell'_{i,q}\rangle$ (if $d'_{j,q} > \ell'_{i,q}$).

Similarly, we consider the at most four pairs $\langle u'_{q,i}, d'_{q,i}\rangle$, $\langle u'_{q,j}, d'_{q,j}\rangle$ and $\langle v'_{q,i}, \ell'_{q,i}\rangle$, $\langle v'_{q,j}, \ell'_{q,j}\rangle$. W.l.o.g. $d'_{q,i} \leq d'_{q,j}$, and so, we set $\langle u_{q,j}, d_{q,j}\rangle = \langle u'_{q,i}, d'_{q,i}\rangle$. Furthermore, $\langle v_{q,j}, \ell_{q,j}\rangle$ can be taken as one of $\langle v'_{q,i}, \ell'_{q,i}\rangle$ and either $\langle u'_{q,j}, d'_{q,j}\rangle$ (if $u'_{q,i} \neq u'_{q,j}$) or $\langle v'_{q,j}, \ell'_{q,j}\rangle$ (if $u'_{q,j} = u'_{q,i}$).

- Otherwise, suppose that we add a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. Let $p, q \notin \{i, j\}$ be arbitrary distinct labels. Clearly, the distances between the vertices in V_p and those in V_q can only be shortened by using some edges of the join. In particular, starting from V_p, if we first reach V_i (resp., V_j) then, since there exists a complete join between V_i and V_j, we can continue with any shortest V_jV_q-path (resp., with any shortest V_iV_q-path). It implies that if $d_{p,q} < d'_{p,q}$ is shortened, then we can choose $u_{p,q}$ as one of $u'_{p,i}$ or $u'_{p,j}$; similarly, if we also have $\ell_{p,q} < \ell'_{p,q}$, then we can choose $v_{p,q}$ as one of $u'_{p,i}, u'_{p,j}, v'_{p,i}, v'_{p,j}$, or (only if $d_{p,q} < d'_{p,q}$) $u'_{p,q}$. Furthermore, we can compute an upper bound on the distance between these above vertices and V_q as follows:

$$\begin{aligned}
dist(u'_{p,i}, V_q) &\leq \min\left\{d'_{p,i} + 1 + d'_{j,q},\ d'_{p,i} + 2 + d'_{i,q}\right\};\\
dist(v'_{p,i}, V_q) &\leq \min\left\{\ell'_{p,i} + 1 + d'_{j,q},\ \ell'_{p,i} + 2 + d'_{i,q}\right\};\\
dist(u'_{p,j}, V_q) &\leq \min\left\{d'_{p,j} + 1 + d'_{i,q},\ d'_{p,j} + 2 + d'_{j,q}\right\};\\
dist(v'_{p,j}, V_q) &\leq \min\left\{\ell'_{p,j} + 1 + d'_{i,q},\ \ell'_{p,j} + 2 + d'_{j,q}\right\}.
\end{aligned}$$

  Note that $d_{p,q} < d'_{p,q}$ only if one of these upper bounds is strictly smaller than $d'_{p,q}$, in which case the latter upper bound is exactly the distance between the corresponding vertex and V_q. Similar arguments apply for $\ell_{p,q}$.

  The above does not work if $p \in \{i, j\}$ for then some of the pairs $\langle u'_{p,i}, d'_{p,i}\rangle$, $\langle v'_{p,i}, \ell'_{p,i}\rangle$ or $\langle u'_{p,j}, d'_{p,j}\rangle$, $\langle v'_{p,j}, \ell'_{p,j}\rangle$ may not exist. However, we can replace the missing pairs by $\langle u'_{p,p}, 0\rangle$, $\langle v'_{p,p}, 0\rangle$ where $u'_{p,p}, v'_{p,p} \in V_p$ are arbitrary vertices. In addition, if $q \in \{i, j\}$ then in order for the above to apply we also need to replace $d'_{q,q}$ by 0.

Note that, as a byproduct of the above rules, we can update $d_{p,q}$ for every distinct labels p and q, after any operation occurs. We are left with doing the same for $d_{p,p}$, for any label p. Clearly, these values cannot change when we add a new (isolated) vertex, with any label, and they can be updated by taking the minimum values when we take the disjoint union of two labeled graphs. We now need to distinguish between the two remaining cases. In what follows, let $d'_{p,p}$ represent the former values.

- Suppose that label i is identified with label j. Then:

$$d_{p,p} = \begin{cases} +\infty & \text{if } p = i\\ \min\left\{d'_{i,i}, d'_{i,j}, d'_{j,j}\right\} & \text{if } p = j\\ d'_{p,p} & \text{else.}\end{cases}$$

- Otherwise, suppose that we add a complete join between the set V_i of vertices labeled i and the set V_j of vertices labeled j. The values $d'_{p,p}$ can only be decreased by using the edges of the join. In particular, if we use a unique edge of the join, then we create a strictly shorter path between $u'_{p,i}$ and $u'_{p,j}$, provided the latter two vertices are different; if they are not, then we create a strictly shorter path either between $u'_{p,i}$ and $v'_{p,j}$ or between $v'_{p,i}$ and $u'_{p,j}$. Otherwise,

we use exactly two edges of the join; we create a strictly shorter path either between $u'_{p,i}$ and $v'_{p,i}$ or between $u'_{p,j}$ and $v'_{p,j}$. Summarizing we have:

$$d_{p,p} = \begin{cases} +\infty & \text{if } p \in \{i,j\},\ |V_p| = 1 \\ \min\{2,\ d'_{p,p}\} & \text{if } p \in \{i,j\},\ |V_p| \geq 2 \\ \min\{d'_{p,p},\ d'_{p,i} + 1 + d'_{j,p},\ d'_{p,i} + 2 + \ell'_{p,i},\ d'_{p,j} + 2 + \ell'_{p,j}\} & \text{if } p \notin \{i,j\},\ u'_{p,i} \neq u'_{p,j} \\ \min\{d'_{p,p},\ d'_{p,i} + 1 + \ell'_{p,j},\ \ell'_{p,i} + 1 + d'_{j,p},\ d'_{p,i} + 2 + \ell'_{p,i}, & \\ \qquad d'_{p,j} + 2 + \ell'_{p,j}\} & \text{else.} \end{cases}$$

□

The bottleneck of the above algorithms is that they require a k-expression as part of the input. So far, the best-known approximation algorithms for clique-width run in $\mathcal{O}(n^3)$-time, that dominates the total running time of our algorithms [88]. However, on a more positive side a k-expression can be computed in linear time for many classes of bounded clique-width graphs. In particular, combining Theorems 2 and 3 with Lemmas 1, 2 and 3 we obtain the following result.

Corollary 4. *For every graph G,* Triangle Counting *and* Girth *can be solved in $\mathcal{O}(k^2 \cdot (n+m))$-time, for every $k \in \{\mathsf{mw}(G), \mathsf{sw}(G), \mathsf{q}(G)\}$.*

4 Parameterization, Hardness and Kernelization for some distance problems on graphs

We prove separability results between clique-width and the upper bounds for clique-width presented in Section 2. More precisely, we consider the problems Diameter, Eccentricities, Hyperbolicity and Betweenness Centrality (defined in Section 4.1), that have already been well studied in the field of "Hardness in P". On the negative side, we show in Section 4.2 that we cannot solve these four above problems with a *fully polynomial* parameterized algorithm, when parameterized by clique-width. However, on a more positive side, we prove the existence of such algorithms in Sections 4.3, 4.4 and 4.5, when parameterized by either the modular-width, the split-width or the P_4-sparseness.

4.1 Distance problems considered

Eccentricity-based problems

The first problem considered is computing the diameter of a graph (maximum length of a shortest-path).

Problem 4 (Diameter)**.**

Input: *A graph $G = (V, E)$.*

Output: *The diameter of G, that is $\max_{u,v \in V} dist_G(u, v)$.*

Hardness results for Diameter have been proved, *e.g.*, in [94, 1, 18, 3, 47].

Our new hardness results are proved for Diameter, while our fully polynomial parameterized algorithms apply to the following more general version of the problem. The *eccentricity* of a given vertex v is defined as $ecc_G(v) = \max_{u \in V} dist_G(u, v)$. Observe that $diam(G) = \max_v ecc_G(v)$.

Problem 5 (Eccentricities)**.**

Input: *A graph* $G = (V, E)$*.*

Output: *The eccentricities of the vertices in* G*, that is* $\max_{u \in V} dist_G(u, v)$ *for every* $v \in V$*.*

Gromov hyperbolicity

Then, we consider the parameterized complexity of computing the Gromov hyperbolicity of a given graph. Gromov hyperbolicity is a measure of how close (locally) the shortest-path metric of a graph is to a tree metric [69]. We refer to [42] for a survey on the applications of Gromov hyperbolicity in computer science.

Problem 6 (Hyperbolicity)**.**

Input: *A graph* $G = (V, E)$*.*

Output: *The hyperbolicity* δ *of* G*, that is:*

$$\max_{u,v,x,y \in V} \frac{dist_G(u, v) + dist_G(x, y) - \max\{dist_G(u, x) + dist_G(v, y), dist_G(u, y) + dist_G(v, y)\}}{2}.$$

Hardness results for Hyperbolicity have been proved in [18, 28, 56]. Some fully polynomial parameterized algorithms, with a different range of parameters than the one considered in this work, have been designed in [49].

Centrality problems

There are different notions of centrality in graphs. For clarity, we choose to keep the focus on one centrality measurement, sometimes called the Betweenness Centrality [57]. More precisely, let $G = (V, E)$ be a graph and let $s, t \in V$. We denote by $\sigma_G(s, t)$ the number of shortest st-paths in G. In particular, for every $v \in V$, $\sigma_G(s, t, v)$ is defined as the number of shortest st-paths passing by v in G.

Problem 7 (Betweenness Centrality)**.**

Input: *A graph* $G = (V, E)$*.*

Output: *The betweenness centrality of every vertex* $v \in V$*, defined as:*

$$BC_G(v) = \sum_{s,t \in V \setminus v} \frac{\sigma_G(s, t, v)}{\sigma_G(s, t)}.$$

See [1, 18, 47] for hardness results on Betweenness Centrality.

4.2 Hardness results for clique-width

The goal in this section is to prove that we cannot solve the problems of Section 4.1 for n-vertex graphs in time $2^{o(cw)}n^{2-\varepsilon}$, for any $\varepsilon > 0$ (Theorems 6—8). These are the first known hardness results for clique-width in the field of "Hardness in P". Our results are conditioned on the Strong Exponential Time Hypothesis (SETH): SAT with N variables cannot be solved in $\mathcal{O}^*(2^{c\cdot N})$-time, for any $c < 1$ [74]. Furthermore, they are derived from similar hardness results obtained for *treewidth*.

Precisely, a *tree decomposition* $(T, \mathcal{X})$ of $G = (V, E)$ is a pair consisting of a tree T and of a family $\mathcal{X} = (X_t)_{t\in V(T)}$ of subsets of V indexed by the nodes of T and satisfying:

- $\bigcup_{t\in V(T)} X_t = V$;
- for any edge $e = \{u, v\} \in E$, there exists $t \in V(T)$ such that $u, v \in X_t$;
- for any $v \in V$, the set of nodes $\{t \in V(T) \mid v \in X_t\}$ induces a subtree, denoted by T_v, of T.

The sets X_t are called *the bags* of the decomposition. Furthermore, $(T, \mathcal{X})$ is termed a *path decomposition* if T is a path. The *width* of a tree decomposition is the size of a largest bag minus one. Finally, the *treewidth* of a graph G, denoted by $\mathsf{tw}(G)$, is the least possible width over its tree decompositions; the *pathwidth* of G, denoted by $\mathsf{pw}(G)$, is the least possible width over its path decompositions.

Several hardness results have already been obtained for treewidth [3]. However, $\mathsf{cw}(G) \leq 2^{\mathsf{tw}(G)}$ for general graphs [27], that does not help us to derive our lower bounds. First, we present a simple and general argument in order to derive such lower bounds. Then, we strengthen our hardness results for the distance problems of Section 4.1 by using relationships between treewidth and clique-width in some graph classes (*i.e.*, bounded-degree graphs [31, 70]).

Our first tool is the following relationship between clique-width and treewidth, that holds for any graph.

Lemma 6 ([13, 48])**.** *For every graph G we have $\mathsf{cw}(G) \leq \mathsf{pw}(G) + 2 \leq \mathcal{O}(\mathsf{tw}(G)\log n)$.*

By Lemma 6, every P-FPT algorithm that runs in $\mathcal{O}(\mathsf{cw}(G)^\alpha \cdot P(n, m))$-time also runs in $\mathcal{O}(\mathsf{tw}(G)^\alpha \cdot P(n, m) \cdot \log^\alpha n)$-time, where $P(n, m)$ is a polynomial function on the graph size. If we require a k-expression as part of the input then such an expression can be computed, for $k = \mathcal{O}(\mathsf{tw}(G)^2 \cdot \log n)$ [2]. In particular, we can always transform a P-FPT algorithm parameterized by clique-width into a P-FPT algorithm parameterized by treewidth.

Conversely, if under SETH we cannot solve some problem Π in $2^{o(k)} \cdot n^{2-\varepsilon}$-time on graphs with pathwidth at most k, for any $\varepsilon > 0$, then under the same assumption we cannot solve Π in $2^{o(k)} \cdot n^{2-\varepsilon}$-time on graphs with *clique-width* at most k. For instance, they proved in [3] that under SETH we cannot decide whether a given graph has diameter either 2 or 3 in $\mathcal{O}(2^{o(\mathsf{pw}(G))} \cdot n^{2-\varepsilon})$-time, for any $\varepsilon > 0$. Therefore, we obtain the following hardness result for clique-width:

[2] We start by computing a tree decomposition $(T, \mathcal{X})$ of the input graph G of width $\mathcal{O}(\mathsf{tw}(G)^2)$. It can be done in $\mathcal{O}(\mathsf{tw}(G)^7 \cdot n \log n)$-time [53]. Then, we compute a path decomposition $(P, \mathcal{Y})$ of the *tree* T, of width $\mathcal{O}(\log n)$, that can be done in time $\mathcal{O}(V(T)) = \mathcal{O}(n)$ [95]. The latter can be transformed into a path decomposition $(P, \mathcal{Y}')$ of G, of width $\mathcal{O}(\mathsf{tw}(G)^2 \log n)$, simply by setting $Y'_s = \bigcup_{t\in Y_s} X_t$ for every $s \in V(P)$. Finally, the proof of the relationship between clique-width and pathwidth in [48] is constructive, and it can be easily transformed into a linear-time algorithm for computing an $\mathcal{O}(\mathsf{tw}(G)^2 \log n)$-expression of G from $(P, \mathcal{Y}')$.

Theorem 5. *Under* SETH*, we cannot solve* DIAMETER *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with clique-width at most* k*, for any* $\varepsilon > 0$*.*

In particular, we cannot decide whether a given graph has diameter either 2 *or* 3 *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with clique-width at most* k*, for any* $\varepsilon > 0$*.*

Then, an interesting question on its own is whether the general hardness results for clique-width still hold for more constrained subclasses of bounded clique-width graphs. Typical such subclasses include the graphs of bounded *linear clique-width* [48], *rank-width* [88], *linear rank-width* [63], *boolean-width* [21] or *linear boolean-width* [90]. The latter parameters for a given graph G are denoted, respectively, by $\mathsf{lin\text{-}cw}(G)$, $\mathsf{rw}(G)$, $\mathsf{lin\text{-}rw}(G)$, $\mathsf{bw}(G)$ and $\mathsf{lin\text{-}bw}(G)$. As shown in Fig. 7, all these parameters stay upper bounded by pathwidth, hence our hardness results still apply in these subcases.

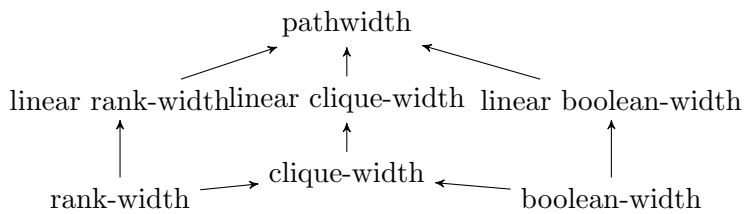

Figure 7: An inclusion diagram of clique-width and some of its relatives.

In what follows, we prove that our hardness results still hold for bounded clique-width graphs of bounded maximum degree. For that, we use another relationship between treewidth and clique-width. Namely:

Lemma 7 ([31, 70])**.** *If* G *has maximum degree at most* d *(with* $d \geq 1$*), we have:*

- $\mathsf{tw}(G) \leq 3d \cdot \mathsf{cw}(G) - 1$*;*
- $\mathsf{cw}(G) \leq 20d \cdot \mathsf{tw}(G) + 22$*.*

Our reductions in what follows are based on Lemma 7, and on previous hardness results for bounded treewidth graphs and bounded-degree graphs [3, 47].

Theorem 6. *Under* SETH*, we cannot solve* DIAMETER *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with maximum degree* 4 *and treewidth at most* k*, for any* $\varepsilon > 0$*.*

In particular, we cannot solve DIAMETER *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with maximum degree* 4 *and clique-width at most* k*, for any* $\varepsilon > 0$*.*

Proof. In [3], Abboud et al. proved that under SETH, we cannot solve DIAMETER in $\mathcal{O}(n^{2-\varepsilon})$-time, for any $\varepsilon > 0$, in the class of tripartite graphs $G = (A \cup C \cup B, E)$ such that: $|A| = |B| = n$, $|C| = \mathcal{O}(\log n)$, and all the edges in E are between C and $A \cup B$. Note that there exists a tree decomposition $(T, \mathcal{X})$ of G such that T is a path and the bags are the sets $\{a\} \cup C,\ a \in A$ and $\{b\} \cup C,\ b \in B$. Hence, $\mathsf{tw}(G) = \mathcal{O}(|C|) = \mathcal{O}(\log n)$ [3].

Then, we use the generic construction of Evald and Dahlgaard [47] in order to transform G into a bounded-degree graph. We prove that graphs with treewidth $\mathcal{O}(\log n)$ can be generated with this construction[3]. More precisely, let T_{big} and T_{small} be rooted balanced binary trees with respective

[3]Our construction has less degrees of freedom than the construction presented in [47].

number of leaves $|A| = |B| = n$ and $|C| = \mathcal{O}(\log n)$. There is a bijective correspondance between the leaves of T_{big} and the vertices in A, resp. between the leaves of T_{big} and the vertices in B. Similarly, there is a bijective correspondance between the leaves of T_{small} and the vertices of C. In order to construct G' from G, we proceed as follows:

- We replace every vertex $u \in A \cup B$ with a disjoint copy T^u_{small} of T_{small}. We also replace every vertex $c \in C$ with two disjoint copies $T^{c,A}_{\text{big}}, T^{c,B}_{\text{big}}$ of T_{big} with a common root.
- For every $a \in A, c \in C$ adjacent in G, we add a path of length p (fixed by the construction) between the leaf of T^a_{small} corresponding to c and the leaf of $T^{c,A}_{\text{big}}$ corresponding to a. In the same way, for every $b \in B, c \in C$ adjacent in G, we add a path of length p between the leaf of T^b_{small} corresponding to c and the leaf of $T^{c,B}_{\text{big}}$ corresponding to b.
- Let T^A_{big} and T^B_{big} be two other disjoint copies of T_{big}. For every $a \in A$ we add a path of length p between the leaf corresponding to a in T^A_{big} and the root of T^a_{small}. For every $b \in B$ we also add a path of length p between the leaf corresponding to b in T^B_{big} and the root of T^b_{small}.
- Finally, for every $u \in A \cup B$, we add a path of length p with one end being the root of T^u_{small}.

The resulting graph G' has maximum degree 4. In [47], they prove that, under SETH, we cannot compute $diam(G')$ in $\mathcal{O}(n^{2-\varepsilon})$-time, for any $\varepsilon > 0$.

We now claim that $tw(G') = \mathcal{O}(\log n)$. Since by Lemma 7 we have $tw(G') = \Theta(cw(G'))$ for bounded-degree graphs, it will imply $cw(G') = \mathcal{O}(\log n)$. In order to prove the claim, we assume w.l.o.g. that the paths added by the above construction have length $p = 1$[4]. Indeed, subdividing an edge does not change the treewidth [15]. Note that in this situation, we can also ignore the pending vertices added for the last step of the construction. Indeed, removing the pending vertices does not change the treewidth either [14]. Hence, from now on we consider the graph G' resulting from the three first steps of the construction by taking $p = 1$.

Let $(T', \mathcal{X}')$ be a tree decomposition of T_{big} of unit width. There is a one-to-one mapping between the nodes $t \in V(T')$ and the edges $e_t \in E(T_{\text{big}})$. Furthermore, let $e^A_t,\ e^B_t, e^{c,A}_t$ and $e^{c,B}_t$ be the copies of edge e_t in the trees $T^A_{\text{big}},\ T^B_{\text{big}},\ T^{c,A}_{\text{big}}$ and $T^{c,B}_{\text{big}},\ c \in C$, respectively. For every node $t \in V(T')$, we define a new bag Y_t as follows. If e_t is not incident to a leaf-node then we set $Y_t = e^A_t \cup e^B_t \cup \left[\bigcup_{c \in C} \left(e^{c,A}_t \cup e^{c,B}_t\right)\right]$. Otherwise, e_t is incident to some leaf-node. Let $a_t \in A,\ b_t \in B$ correspond to the leaf. We set $Y_t = V\left(T^{a_t}_{\text{small}}\right) \cup V\left(T^{b_t}_{\text{small}}\right) \cup e^A_t \cup e^B_t \cup \left[\bigcup_{c \in C} \left(e^{c,A}_t \cup e^{c,B}_t\right)\right]$. By construction, $(T', (Y_t)_{t \in V(T')})$ is a tree decomposition of G'. In particular, $tw(G') \leq \max_{t \in V(T')} |Y_t| = \mathcal{O}(\log n)$, that finally proves the claim.

Finally, suppose by contradiction that $diam(G')$ can be computed in $2^{o(tw(G'))} \cdot n^{2-\varepsilon}$-time, for some $\varepsilon > 0$. Since $tw(G') = \mathcal{O}(\log n)$, it implies that $diam(G')$ can be computed in $\mathcal{O}(n^{2-\varepsilon})$-time, for some $\varepsilon > 0$. The latter refutes SETH. Hence, under SETH we cannot solve Diameter in $2^{o(k)} \cdot n^{2-\varepsilon}$-time on graphs with maximum degree 4 and treewidth at most k, for any $\varepsilon > 0$. This negative result also holds for clique-width since $cw(G') = \Theta(tw(G'))$. □

The following reduction to Betweenness Centrality is from [47]. Our main contribution is to upper bound the clique-width and the treewidth of their construction.

[4]The hardness result of [47] holds for $p = \omega(\log n)$. We reduce to the case $p = 1$ *only* for computing the treewidth.

Theorem 7. *Under* SETH*, we cannot solve* Betweenness Centrality *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with maximum degree* 4 *and treewidth at most* k*, for any* $\varepsilon > 0$*.*

In particular, we cannot solve Betweenness Centrality *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with maximum degree* 4 *and clique-width at most* k*, for any* $\varepsilon > 0$*.*

Proof. Let G' be the graph from the reduction of Theorem 6. In [47], the authors propose a reduction from G' to H such that, under SETH, we cannot solve Betweenness Centrality for H in $\mathcal{O}(n^{2-\varepsilon})$-time, for any $\varepsilon > 0$. In order to prove the theorem, it suffices to prove $tw(H) = \Theta(tw(G'))$. Indeed, the construction of H from G' is as follows.

- For every $u \in A \cup B$, we remove the path of length p with one end being the root of T^u_{small}, added at the last step of the construction of G'. This operation can only decrease the treewidth.
- Then, we add a path of length p between the respective roots of T^A_{big} and T^B_{big}. Recall that we can assume $p = 1$ since subdividing an edge does not modify the treewidth [15]. Adding an edge to a graph increases its treewidth by at most one.
- Finally, let H_1 be the graph so far constructed. We make a disjoint copy H_2 of H_1. This operation does not modify the treewidth. Then, for every $b \in B$, let ${T^b_{\text{small}}}'$ be the copy of T^b_{small} in H_2. We add a new vertex b' that is uniquely adjacent to the root of ${T^b_{\text{small}}}'$. The addition of pending vertices does not modify the treewidth either [14].

Overall, $tw(H) \leq tw(G') + 1 = \mathcal{O}(\log n)$. Furthermore, since H has maximum degree at most 4, by Lemma 7 we have $cw(H) = \Theta(tw(H)) = \mathcal{O}(\log n)$. □

Our next reduction for Hyperbolicity is inspired from the one presented in [18]. However, the authors in [18] reduce from a special case of Diameter where we need to distinguish between graphs with diameter either 2 or 3. In order to reduce from a more general case of Diameter we need to carefully refine their construction.

Theorem 8. *Under* SETH*, we cannot solve* Hyperbolicity *in* $2^{o(k)} \cdot n^{2-\varepsilon}$*-time on graphs with clique-width and treewidth at most* k*, for any* $\varepsilon > 0$*.*

Proof. We use the graph G' from the reduction of Theorem 6. More precisely, let us take $p = \omega(\log n)$ for the size of the paths in the construction [5]. It has been proved in [47] that either $diam(G') = (4 + o(1))p$ or $diam(G') = (6 + o(1))p$. Furthermore, under SETH we cannot decide in which case we are in truly subquadratic time.

Our reduction is inspired from [18]. Let H be constructed from G' as follows (see also Fig. 8).

- We add two disjoint copies V_x, V_y of $V(G')$ and the three vertices $x, y, z \notin V(G')$. We stress that V_x and V_y are independent sets. Furthermore, for every $v \in V$, we denote by v_x and v_y the copies of v in V_x and V_y, respectively.
- For every $v \in V(G')$, we add a vv_x-path P^x_v of length $(3/2 + o(1))p$, and similarly we add a vv_y-path P^y_v of length $(3/2 + o(1))p$.
- Furthermore, for every $v \in V(G')$ we also add a xv_x-path Q^x_v of length $(3/2 + o(1))p$; a yv_y-path Q^y_v of length $(3/2 + o(1))p$; a zv_x-path $Q^{z,x}_v$ of length $(3/2 + o(1))p$ and a zv_y-path $Q^{z,y}_v$ of length $(3/2 + o(1))p$.

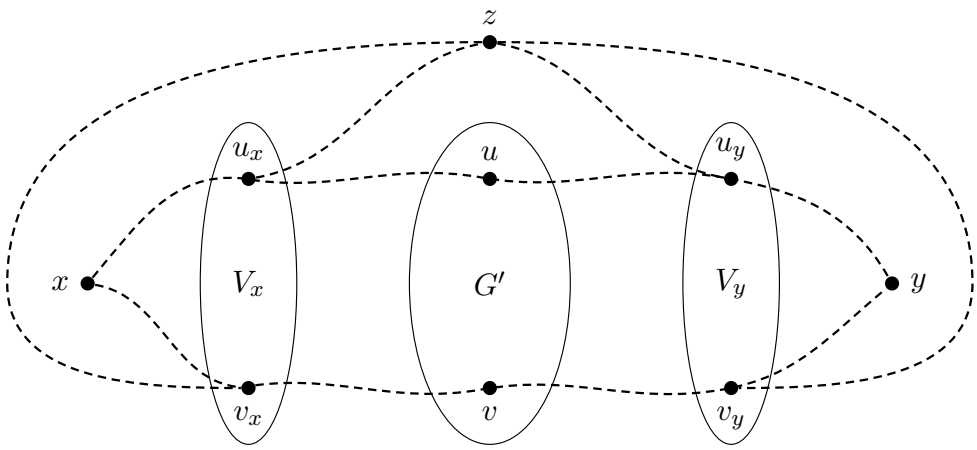

Figure 8: The graph H from the reduction of Theorem 8. Dashed lines corresponds to paths.

We claim that the resulting graph H is such that $tw(H) = tw(G') + \mathcal{O}(1)$ and $cw(H) = cw(G') + \mathcal{O}(1)$. Indeed, let us first consider $H' = H \setminus \{x, y, z\}$. The graph H' is obtained from G' by adding some disjoint trees rooted at the vertices of $V(G')$. In particular, it implies $tw(H') = tw(G')$, hence (by adding x, y, z in every bag) $tw(H) \leq tw(H') + 3 \leq tw(G') + 3$.

Furthermore, let us fix a k-expression for G'. We transform it to a $(k + 16)$-expression for H as follows. We start adding x, y, z with three distinct new labels. In particular, we now use $k + 3$ labels. Then, we follow the k-expression for G'. Suppose a new vertex $v \in V(G')$, with label i is introduced. It corresponds to some tree T_v in H', that is rooted at v. Every such a tree has clique-width at most 3 [68]. So, as an intermediate step, let us fix a 3-expression for T_v. We transform it to a 12-expression for T_v: with each new label encoding the former label in the 3 expression (3 possibilities), and whether the node is either the root v or adjacent to one of x, y, z (4 possibilities). This way, we can make x, y, z adjacent to their neighbours in T_v, using the join operation. Then, since the root v has a distinguished label, we can "freeze" all the other nodes in $T_v \setminus v$ using an additional new label and relabeling operations. Overall, we use $12 + 1 = 13$ new labels during this step, *i.e.*, the labels for the 12-expression of T_v plus an additional freezing label. Finally, we relabel v with its original label i in the k-expression of G', and then we continue following this k-expression. Note that doing so, we can reuse the same 13 intermediate labels for every tree T_v. Therefore, we only use $k + 3 + 13 = k + 16$ labels. Summarizing, $cw(H) \leq cw(G') + 16$.

Next, we claim that $\delta(H) \geq (3 + o(1))p$ if $diam(G') = (6 + o(1))p$, while $\delta(H) \leq (11/4 + o(1))p$ if $diam(G') = (4 + o(1))p$. Recall that by Theorem 6, under SETH we cannot decide in which case we are in time $2^{o(tw(G'))}n^{2-\varepsilon} = 2^{o(cw(G'))}n^{2-\varepsilon}$, for any $\varepsilon > 0$. Therefore, proving the claim will prove the theorem.

Firstly, suppose that $diam(G') = (6+o(1))p$. Let $u, v \in V(G')$ satisfy $dist_{G'}(u, v) = (6+o(1))p$. Observe that $diam(G') \leq (6 + o(1))p = 4 \cdot (3/2 + o(1))p$, therefore G' is an isometric subgraph of H by construction. Then, $S_1 = dist_H(u, v) + dist_H(x, y) = (12 + o(1))p$; $S_2 = dist_H(u, x) + dist_H(v, y) = (6 + o(1))p$; $S_3 = dist_H(u, y) + dist_H(v, x) = S_2$. As a result, we obtain $\delta(H) \geq (S_1 - \max\{S_2, S_3\})/2 = (3 + o(1))p$.

Secondly, suppose that $diam(G') = (4 + o(1))p$. We want to prove $\delta(H) \leq (11/4 + o(1))p$. By

[5]We stress that unlike treewidth, the hyperbolicity of a graph can be increased after an edge subdivision.

contradiction, let $a, b, c, d \in V(H)$ satisfy:

$$S_1 = dist_H(a,b) + dist_H(c,d) \geq S_2 = dist_H(a,c) + dist_H(b,d) \geq S_3 = dist_H(a,d) + dist_H(b,c),$$

$$S_1 - S_2 > (11/2 + o(1))p.$$

The hyperbolicity of a given 4-tuple is upper bounded by the minimum distance between two vertices of the 4-tuple [17, 24, 96]. So, let us consider the distances in H.

- Let $v \in V(G')$. For every $u \in V(G')$, $dist_H(u,v) \leq dist_G(u,v) \leq (4 + o(1))p$.

  Furthermore for every $u' \in P_u^x$, $dist_H(v,u') \leq dist_H(v,u) + dist_H(u,u') \leq (11/2 + o(1))p$. Similarly for every $u' \in P_u^y$, $dist_H(v,u') \leq dist_H(v,u) + dist_H(u,u') \leq (11/2 + o(1))p$.

  For every $u' \in Q_u^x$, $dist_H(v,u') \leq dist_H(v,x) + dist_H(x,u') \leq (9/2 + o(1))p$. We prove in the same way that for every $u' \in Q_u^y \cup Q_u^{z,x} \cup Q_u^{z,y}$, $dist_H(v,u') \leq (9/2 + o(1))p$.

  Summarizing, $ecc_H(v) \leq (11/2 + o(1))p$.

- Let $v' \in P_v^x$, for some $v \in V(G')$.

  For every $u \in V(G')$ and $u' \in P_u^x$ there are two cases. Suppose that $dist_H(u', u_x) \leq p + o(p)$ or $dist_H(v', v_x) \leq p + o(p)$. Then, $dist_H(v', u') \leq dist_H(v', v_x) + dist_H(v_x, u_x) + dist_H(u_x, u') \leq (1 + 3 + 3/2 + o(1))p = (11/2 + o(1))p$. Otherwise, $\max\{dist_H(u', u), dist_H(v', v)\} \leq (1/2 + o(1))p$, and so, $dist_H(u', v') \leq dist_H(u', u) + dist_H(u, v) + dist_H(v', v) \leq (5 + o(1))p$. Similarly (replacing u_x with u_y), for every $u' \in P_u^y$ we have $dist(v', u') \leq (11/2 + o(1))p$.

  For every $u' \in Q_u^x$, $dist_H(v', u') \leq dist_H(v', v_x) + dist_H(x, v_x) + dist_H(x, u') \leq (9/2 + o(1))p$. In the same way for every $u' \in Q_u^{z,x} \cup Q_u^{z,y}$, $dist_H(v', u') \leq dist_H(v', v_x) + dist_H(z, v_x) + dist_H(z, u') \leq (9/2 + o(1))p$.

  For every $u' \in Q_u^y$, we first need to observe that $dist_H(v_x, u_y) = (3 + o(1))p$ and $dist_H(v, y) = (3 + o(1))p$. In particular if $dist_H(v, v') \leq p + o(p)$ then, $dist_H(v', u') \leq dist_H(v', v) + dist_H(v, y) + dist_H(y, u') \leq (11/2 + o(1))p$. Otherwise, $dist_H(v', u') \leq dist_H(v', v_x) + dist_H(v_x, u_y) + dist_H(u_y, u') \leq (1/2 + 3 + 3/2 + o(1))p = (5 + o(1))p$.

  Summarizing, $ecc_H(v') \leq (11/2 + o(1))p$.

- Let $v' \in P_v^y$, for some $v \in V(G')$. We prove $ecc_H(v') \leq (11/2 + o(1))p$ in the same way as above.

- Let $v' \in Q_v^{z,x} \cup Q_v^{z,y}$, for some $v \in V(G')$.

  For every $u \in V(G')$ and for every $u' \in Q_u^{z,x} \cup Q_u^{z,y}$ we have $dist_H(v', u') \leq dist_H(v', z) + dist_H(z, u') \leq (3 + o(1))p$.

  For every $u' \in Q_u^x$ we have $dist_H(v', u') \leq dist_H(v', z) + dist_H(z, u_x) + dist_H(u_x, u') \leq (9/2 + o(1))p$. Similarly for every $u' \in Q_u^y$ we have $dist_H(v', u') \leq (9/2 + o(1))p$.

  Summarizing, $ecc_H(v') \leq (11/2 + o(1))p$.

In particular, every vertex in H has eccentricity at most $(11/2 + o(1))p$, except maybe those in $\bigcup_{v \in V(G')} Q_v^x = X$ and those in $\bigcup_{v \in V(G')} Q_v^y = Y$. However, $S_1 - S_2 \leq \min\{dist_H(a,b), dist_H(c,d)\}$ [24]. So, we can assume w.l.o.g. $a, c \in X$ and $b, d \in Y$. Furthermore, $S_1 - S_2 \leq 2 \cdot dist_H(a,c)$ [17, 96]. Hence, $(11/2 + o(1))p < S_1 - S_2 \leq 2 \cdot dist_H(a,c) \leq 2 \cdot (dist_H(a,x) +$

$dist_H(c,x)$). It implies $\max\{dist_H(a,x), dist_H(c,x)\} > (11/8+o(1))p = (3/2-1/8+o(1))p$. Assume by symmetry that $dist_H(a,x) > (3/2-1/8+o(1))p$. Then, $dist_H(a,V_x) < (1/8+o(1))p$. However, $dist_H(a,b) \leq dist_H(a,V_x) + (3+o(1))p + dist_H(b,V_y) < (1/8+3+3/2+o(1))p < (11/2+o(1))p$. A contradiction. Therefore, we obtain as claimed that $\delta(H) \leq (11/4+o(1))p$. □

It is open whether any of these above problems can be solved in time $2^{\mathcal{O}(k)} \cdot n$ on graphs with clique-width at most k (resp., on graphs with treewidth at most k, see [3, 73]).

4.3 Parameterized algorithms with split decomposition

We now show how to use split decomposition as an efficient preprocessing method for Diameter, Eccentricities, Hyperbolicity and Betweenness Centrality. Improvements obtained with modular decomposition will be discussed in Section 4.4. Roughly, we show that in order to solve the problems considered, it suffices to solve some *weighted* variant of the original problem for every split component (subgraphs of the split decomposition) separately. However, weights intuitively represent the remaining of the graph, so, we need to account for some dependencies between the split components in order to define the weights properly.

In order to overcome this difficulty, we use in what follows a tree-like structure over the split components in order to design our algorithms. A *split decomposition tree* of G is a labeled tree T where the nodes are in bijective correspondance with the subgraphs of the split decomposition of G, and the edges of T are in bijective correspondance with the simple decompositions used for their computation.

More precisely:

- If G is either degenerate, or prime for split decomposition, then T is reduced to a single node that is either labeled star, complete or prime;
- Otherwise, let (A,B) be a split of G and let $G_A = (A \cup \{b\}, E_A),\ G_B = (B \cup \{a\}, E_B)$ be the corresponding subgraphs of G. We construct the split decomposition trees T_A, T_B for G_A and G_B, respectively. Furthermore, the split marker vertices a and b are contained in a unique split component of G_A and G_B, respectively. We obtain T from T_A and T_B by adding an edge between the two nodes that correspond to these subgraphs.

A split decomposition tree can be constructed in linear time [22]. Furthermore we stress that we can use the labels in this tree in order to decide in *constant time* whether a given split component is either a star, complete or prime. Therefore, in what follows, we do not take into account this preliminary test in our algorithms.

Diameter and Eccentricities

Lemma 8. *Let (A,B) be a split of $G=(V,E)$ and let $G_A = (A \cup \{b\}, E_A),\ G_B = (B \cup \{a\}, E_B)$ be the corresponding subgraphs of G. Then, for every $u \in A$ we have:*

$$ecc_G(u) = \max\{ecc_{G_A}(u), dist_{G_A}(u,b) + ecc_{G_B}(a) - 1\}.$$

Proof. Let $C = N_G(B) \subseteq A$ and $D = N_G(A) \subseteq B$. In order to prove the claim, we first need to observe that, since (A,B) is a split of G, we have, for every $v \in V$:

$$dist_G(u,v) = \begin{cases} dist_{G_A}(u,v) \text{ if } v \in A \\ dist_G(u,C) + 1 + dist_G(v,D) \text{ if } v \in B. \end{cases}$$

Furthermore, $dist_G(u, C) = dist_{G_A}(u, b) - 1$, and similarly $dist_G(v, D) = dist_{G_B}(v, a) - 1 \leq ecc_{G_B}(a) - 1$. Hence, $ecc_G(u) \leq \max\{ecc_{G_A}(u), dist_{G_A}(u, b) + ecc_{G_B}(a) - 1\}$.

Conversely, $ecc_{G_A}(u) = \max\{dist_{G_A}(u, b)\} \cup \{dist_{G_A}(u, v) \mid v \in A\} = \max\{dist_G(u, D)\} \cup \{dist_G(u, v) \mid v \in A\} \leq ecc_G(u)$. In the same way, let $v \in B$ maximize $dist_G(v, C)$. We have: $dist_G(u, v) = dist_{G_A}(u, b) + ecc_{G_B}(a) - 1 \leq ecc_G(u)$. □

Theorem 9. *For every graph G,* Eccentricities *can be solved in $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$-time.*

In particular, Diameter *can be solved in $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$-time.*

Proof. Let T be a split decomposition tree of G, with its nodes being in bijective correspondance with the split components $C_1, C_2, \ldots, C_k$. It can be computed in linear time [22]. We root T in C_1. For every $1 \leq i \leq k$, let T_i be the subtree of T that is rooted in C_i. If $i > 1$ then let $C_{p(i)}$ be its parent in T. By construction of T, the edge $\{C_{p(i)}, C_i\} \in E(T)$ corresponds to a split (A_i, B_i) of G. Let $G_{A_i} = (A_i \cup \{b_i\}, E_{A_i})$, $G_{B_i} = (B_i \cup \{a_i\}, E_{B_i})$ be the corresponding subgraphs of G. W.l.o.g. $C_i \subseteq G_{A_i}$. We observe that: T_i is a split decomposition tree of G_{A_i}, and similarly $T \setminus T_i$ is a split decomposition tree of G_{B_i}.

Our algorithm proceeds in two main steps, with each step corresponding to a different traversal of the tree T. First, let $G_1 = G$ and let $G_i = G_{A_i}$ for every $i > 1$. We first compute, for every $1 \leq i \leq k$ and for every $v_i \in V(C_i)$, its eccentricity in G_i. In order to do so, we proceed by dynamic programming on the tree T:

- If C_i is a leaf of T then Eccentricities can be solved: in $\mathcal{O}(|V(C_i)|)$-time if C_i induces a star or a complete graph; and in $\mathcal{O}(|V(C_i)|^3) = \mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_i)|)$-time else.

- Otherwise C_i is an internal node of T. Let $C_{i_1}, C_{i_2}, \ldots, C_{i_l}$ be the children of C_i in T. Every edge $\{C_i, C_{i_t}\} \in E(T)$, $1 \leq t \leq l$ corresponds to a split (A_{i_t}, B_{i_t}) of G_i, where $C_{i_t} \subseteq G_{A_{i_t}} = G_{i_t}$. We name $b_{i_t} \in V(C_{i_t})$, $a_{i_t} \in V(C_i)$ the vertices added after the simple decomposition. Furthermore, let us define $e(a_{i_t}) = ecc_{G_{i_t}}(b_{i_t}) - 1$. For every other vertex $u \in V(C_i) \setminus \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$, we define $e(u) = 0$. Then, applying Lemma 8 for every split (A_{i_t}, B_{i_t}) we get the following equality:

$$\forall u \in V(C_i),\ ecc_{G_i}(u) = \max_{v \in V(C_i)} dist_{C_i}(u, v) + e(v).$$

  We distinguish between three cases.

  1. If C_i is complete, then we need to find $x_i \in V(C_i)$ maximizing $e(x_i)$, and $y_i \in V(C_i) \setminus \{x_i\}$ maximizing $e(y_i)$. It can be done in $\mathcal{O}(|V(C_i)|)$-time. Furthermore, for every $u \in V(C_i)$, we have $ecc_{G_i}(u) = 1 + e(x_i)$ if $u \neq x_i$, and $ecc_{G_i}(x_i) = \max\{e(x_i), 1 + e(y_i)\}$.
  2. If C_i is a star with center node r, then we need to find a leaf $x_i \in V(C_i) \setminus \{r\}$ maximizing $e(x_i)$, and another leaf $y_i \in V(C_i) \setminus \{x_i, r\}$ maximizing $e(y_i)$. It can be done in $\mathcal{O}(|V(C_i)|)$-time. Furthermore, $ecc_{G_i}(r) = \max\{e(r), 1 + e(x_i)\}$, $ecc_{G_i}(x_i) = \max\{e(x_i), 1 + e(r), 2 + e(y_i)\}$, and for every other $u \in V(C_i) \setminus \{x_i, r\}$ we have $ecc_{G_i}(u) = \max\{1 + e(r), 2 + e(x_i)\}$.
  3. Otherwise, we have $|V(C_i)| \leq \mathsf{sw}(G)$, and so all the eccentricities can be computed in $\mathcal{O}(|V(C_i)||E(C_i)|) = \mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_i)|)$-time.

Overall, this step takes total time $\mathcal{O}(\mathsf{sw}(G)^2 \cdot \sum_i |V(C_i)|) = \mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$. Furthermore, since $G_1 = G$, we have computed $ecc_G(v_1)$ for every $v_1 \in V(C_1)$.

Second, for every $2 \leq i \leq k$, we recall that by Lemma 8:

$$\forall v_i \in V(G_i),\ ecc_G(v_i) = \max\{ecc_{G_i}(v_i), dist_{G_i}(v_i, b_i) + ecc_{G_{B_i}}(a_i) - 1\}.$$

In particular, since we have already computed $ecc_{G_i}(v_i)$ for every $v_i \in V(C_i)$ (and as a byproduct, $dist_{G_i}(v_i, b_i)$), we can compute $ecc_G(v_i)$ from $ecc_{G_{B_i}}(a_i)$. So, we are left to compute $ecc_{G_{B_i}}(a_i)$ for every $2 \leq i \leq k$. In order to do so, we proceed by reverse dynamic programming on the tree T.

More precisely, let $C_{p(i)}$ be the parent node of C_i in T, and let $C_{j_0} = C_i, C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ denote the children of $C_{p(i)}$ in T. For every $0 \leq t \leq k$, the edge $\{C_{p(i)}, C_{j_t}\}$ represents a split (A_{j_t}, B_{j_t}), where $C_{j_t} \subseteq G_{A_{j_t}} = G_{j_t}$. So, there has been vertices $b_{j_t} \in V(C_{j_t}),\ a_{j_t} \in V(C_{p(i)})$ added by the corresponding simple decomposition. We define $e'(a_{j_t}) = ecc_{G_{j_t}}(b_{j_t}) - 1$. Furthermore, if $p(i) > 1$, let $C_{p^2(i)}$ be the parent of $C_{p(i)}$ in T. Again, the edge $\{C_{p^2(i)}, C_{p(i)}\}$ represents a split $(A_{p(i)}, B_{p(i)})$, where $C_{p(i)} \subseteq G_{A_{p(i)}} = G_{p(i)}$. So, there have been vertices $b_{p(i)} \in V(C_{p(i)}),\ a_{p(i)} \in V(C_{p^2(i)})$ added by the corresponding simple decomposition. Let us define $e'(b_{p(i)}) = ecc_{G_{B_{p(i)}}}(a_{p(i)}) - 1$ (obtained by reverse dynamic programming on T). Finally, for any other vertex $u \in V(C_{p(i)})$, let us define $e'(u) = 0$. Then, by applying Lemma 8 it comes:

$$\forall 0 \leq t \leq k,\ ecc_{G_{B_{i_t}}}(a_{i_t}) = \max_{v \in V(C_{p(i)}) \setminus \{a_{i_t}\}} dist_{C_{p(i)}}(a_{i_t}, v) + e'(v).$$

We can adapt the techniques of the first step in order to compute all the above values in $\mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_{p(i)})|)$-time. Overall, the time complexity of the second step is also $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$.

Finally, since a split decomposition can be computed in $\mathcal{O}(n+m)$-time, and all of the subsequent steps take $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$-time, the total running time of our algorithm is an $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$. □

Gromov hyperbolicity

It has been proved in [96] that for every graph G, if every split component of G is δ-hyperbolic then $\delta(G) \leq \max\{1, \delta\}$. We give a self-contained proof of this result, where we characterize the gap between $\delta(G)$ and the maximum hyperbolicity of its split components.

Lemma 9. *Let (A, B) be a split of $G = (V, E)$ and let $C = N_G(B) \subseteq A,\ D = N_G(A) \subseteq B$. Furthermore, let $G_A = (A \cup \{b\}, E_A),\ G_B = (B \cup \{a\}, E_B)$ be the corresponding subgraphs of G.*

Then, $\delta(G) = \max\{\delta(G_A), \delta(G_B), \delta^*\}$ where:

$$\delta^* = \begin{cases} 1 & \text{if neither } C \text{ nor } D \text{ is a clique;} \\ 1/2 & \text{if } \min\{|C|, |D|\} \geq 2 \text{ and exactly one of } C \text{ or } D \text{ is a clique;} \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Since G_A, G_B are isometric subgraphs of G, we have $\delta(G) \geq \max\{\delta(G_A), \delta(G_B)\}$. Conversely, for every $u, v, x, y \in V$ define L and M to be the two largest sums amongst $\{dist_G(u, v) + dist_G(x, y), dist_G(u, x) + dist_G(v, y), dist_G(u, y) + dist_G(v, x)\}$. Write $\delta(u, v, x, y) = (L - M)/2$. Furthermore, assume that $\delta(u, v, x, y) = \delta(G)$. W.l.o.g., $|\{u, v, x, y\} \cap A| \geq |\{u, v, x, y\} \cap B|$. In particular, if $u, v, x, y \in A$ then $\delta(u, v, x, y) \leq \delta(G_A)$. Otherwise, there are two cases.

- Suppose $|\{u,v,x,y\} \cap A| = 3$. W.l.o.g., let $y \in B$. Then, for every $w \in \{u,v,x\}$ we have $dist_G(w,y) = dist_{G_A}(w,b) + dist_{G_B}(a,y) - 1$. Hence, $\delta(u,v,x,y) = \delta(u,v,x,b) \leq \delta(G_A)$.

- Otherwise, $|\{u,v,x,y\} \cap A| = 2$. W.l.o.g., let $x,y \in B$. Observe that $M = dist_G(u,x) + dist_G(v,y) = dist_G(u,y) + dist_G(v,x) = dist_{G_A}(u,b) + dist_{G_A}(v,b) + dist_{G_B}(a,x) + dist_{G_B}(a,y) - 2$. Furthermore, $L = dist_G(u,v) + dist_G(x,y) \leq dist_{G_A}(u,b) + dist_{G_A}(v,b) + dist_{G_B}(a,x) + dist_{G_B}(a,y)$. Hence, $\delta(u,v,x,y) = \max\{0, L-M\}/2 \leq 1$. In particular:

  - Suppose $\min\{|C|,|D|\} = 1$. Then, the 4-tuple u,v,x,y is disconnected by some cut-vertex c. In particular, $M = dist_G(u,c) + dist_G(v,c) + dist_G(c,x) + dist_G(c,y) \geq L$, and so, $\delta(u,v,x,y) = 0$. Thus we assume from now on that $\min\{|C|,|D|\} \geq 2$.
  - Suppose $L - M = 2$. It implies both a is on a shortest xy-path (in G_B) and b is on a shortest uv-path (in G_A). Since there can be no simplicial vertices on a shortest path, we obtain that neither a nor b can be simplicial. Thus, C and D are not cliques. Conversely, if C and D are not cliques then there exists an induced C_4 with two ends in C and two ends in D. Since $\delta(C_4) = 1$, we get $\delta(G) \geq 1$.
  - Suppose $L - M = 1$. Either C or D is not a clique. Conversely, if either C or D is not a clique then, since we also assume $\min\{|C|,|D|\} \geq 2$, there exists either an induced C_4 or an induced diamond with two vertices in C and two vertices in D. As a result, $\delta(G) \geq 1/2$.

□

Theorem 10. *For every graph G,* Hyperbolicity *can be solved in $\mathcal{O}(\mathsf{sw}(G)^3 \cdot n + m)$-time.*

Proof. First we compute in linear time the split components $C_1, C_2, \ldots, C_k$ of G. By Lemma 9, we have $\delta(G) \geq \max_i \delta(C_i)$. Furthermore, for every $1 \leq i \leq k$ we have: if C_i induces a star or a complete graph, then $\delta(C_i) = 0$; otherwise, $|V(C_i)| \leq \mathsf{sw}(G)$, and so, $\delta(C_i)$ can be computed in $\mathcal{O}(|V(C_i)|^4) = \mathcal{O}(\mathsf{sw}(G)^3 \cdot |V(C_i)|)$-time, simply by iterating over all possible 4-tuples. Summarizing, we can compute $\max_i \delta(C_i)$ in $\mathcal{O}(\mathsf{sw}(G)^3 \cdot \sum_i |V(C_i)|) = \mathcal{O}(\mathsf{sw}(G)^3 \cdot n)$-time. By Lemma 9 we have $\delta(G) \leq \max\{1, \max_i \delta(C_i)\}$. Therefore, if $\max_i \delta(C_i) \geq 1$ then we are done. Otherwise, in order to compute $\delta(G)$, by Lemma 9 it suffices to check whether the sides of every split used for the split decomposition induce a complete subgraph. For that, we use a split decomposition tree T of G. Indeed, recall that the edges of T are in bijective correspondance with the splits.

Let us root T in C_1. Notations are from the proof of Theorem 9. In particular, for every $1 \leq i \leq k$ let T_i be the subtree of T that is rooted in C_i. If $i > 1$ then let $C_{p(i)}$ be its parent in T. By construction of T, the edge $\{C_{p(i)}, C_i\} \in E(T)$ corresponds to a split (A_i, B_i) of G. Let $G_{A_i} = (A_i \cup \{b_i\}, E_{A_i})$, $G_{B_i} = (B_i \cup \{a_i\}, E_{B_i})$ be the corresponding subgraphs of G. W.l.o.g. $C_i \subseteq G_{A_i}$. Furthermore, vertex a_i is simplicial in G_{B_i} if and only if the side $N_G(A_i)$ is a clique. Similarly, vertex b_i is simplicial in G_{A_i} if and only if the side $N_G(B_i)$ is a clique. So, we perform tree traversals of T in order to decide whether a_i and b_i are simplicial.

More precisely, we recall that T_i and $T \setminus T_i$ are split decomposition trees of G_{A_i} and G_{B_i}, respectively. We now proceed in two main steps.

- First, we decide whether b_i is simplicial in G_{A_i} by dynamic programming. More precisely, let $C_{i_1}, C_{i_2}, \ldots, C_{i_k}$ be the children of C_i in T. (possibly, $k = 0$ if C_i is a leaf). Then, b_i is simplicial in G_{A_i} if and only if: it is simplicial in C_i; and for every $1 \leq t \leq k$ such that

$\{b_i, a_{i_t}\} \in E(C_i)$, we have that b_{i_t} is simplicial in $G_{A_{i_t}}$. In particular, testing whether b_i is simplicial in C_i takes time: $\mathcal{O}(1)$ if C_i induces a star (as we only need to check whether b_i is the center); $\mathcal{O}(1)$ if C_i induces a complete graph (as then the answer is trivially true); and $\mathcal{O}(|V(C_i)|^2) = \mathcal{O}(\mathsf{sw}(G) \cdot |V(C_i)|)$ otherwise. Since a vertex can have at most $|V(C_i)| - 1$ neighbours in C_i, testing whether b_i is simplicial in G_{A_i} can be done in $\mathcal{O}(|V(C_i)|)$ additional time. So, overall, the first step takes $\mathcal{O}(\mathsf{sw}(G) \cdot \sum_i |V(C_i)|) = \mathcal{O}(\mathsf{sw}(G) \cdot n)$-time.

- Second, we decide whether a_i is simplicial in G_{B_i} by reverse dynamic programming. Let $C_{j_0} = C_i, C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ denote the children of $C_{p(i)}$ in T. Furthermore, if $p(i) \neq 1$ then let $C_{p^2(i)}$ be the parent of $C_{p(i)}$ in T. Then, a_i is simplicial in G_{B_i} if and only if: it is simplicial in $C_{p(i)}$; for every $1 \leq t \leq k$ such that $\{a_i, a_{j_t}\} \in E(C_{p(i)})$, we have that b_{j_t} is simplicial in $G_{A_{j_t}}$; if $p(i) \neq 1$ and $\{a_i, b_{p(i)}\} \in E(C_{p(i)})$, we also have that $a_{p(i)}$ is simplicial in $G_{B_{p(i)}}$. Testing, for every $0 \leq t \leq k$, whether a_{j_t} is simplicial in $C_{p(i)}$ takes total time: $\mathcal{O}(|V(C_{p(i)})|)$ if $C_{p(i)}$ induces a star or a complete graph; and $\mathcal{O}(|V(C_{p(i)})|^3) = \mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_{p(i)})|)$ otherwise.

  Then, for stars and prime components, we can test, for every $0 \leq t \leq k$, whether a_{j_t} is simplicial in $G_{B_{j_t}}$ in total $\mathcal{O}(|E(C_{p(i)})|)$-time, that is $\mathcal{O}(|V(C_{p(i)})|)$ for stars and $\mathcal{O}(|V(C_{p(i)})|^2) = \mathcal{O}(\mathsf{sw}(G) \cdot |V(C_{p(i)})|)$ for prime components. For the case where $C_{p(i)}$ is a complete graph then, since all the vertices in $C_{p(i)}$ are pairwise adjacent, we only need to check whether there is at least one vertex a_{j_t} such that b_{j_t} is non simplicial in $G_{A_{j_t}}$, and also if $p(i) > 1$ whether $a_{p(i)}$ is non simplicial in $G_{B_{p(i)}}$. It takes $\mathcal{O}(|V(C_{p(i)})|)$-time.

  So, overall, the second step takes $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$-time.

□

Corollary 11 ([96]). *For every connected graph G we have $\delta(G) \leq \max\{1, \lfloor(\mathsf{sw}(G) - 1)/2\rfloor\}$.*

Betweenness Centrality

The following subsection can be seen as a broad generalization of the preprocessing method presented in [89]. We start introducing a generalization of Betweenness Centrality for *vertex-weighted* graphs. Admittedly, the proposed generalization is somewhat technical. However, it will make easier the dynamic programming of Theorem 12.

Precisely, let $G = (V, E, \alpha, \beta)$ with $\alpha, \beta : V \to \mathbb{N}$ be weight functions. Intuitively, for a split marker vertex v, $\alpha(v)$ represents the side of the split replaced by v, while $\beta(v)$ represents the total number of vertices removed by the simple decomposition. For every path $P = (v_1, v_2, \ldots, v_\ell)$ of G, the *length* of P is equal to the number $\ell - 1$ of edges in the path, while the *cost* of P is equal to $\prod_{i=1}^{\ell} \alpha(v_i)$. Furthermore, for every $s, t \in V$, the value $\sigma_G(s, t)$ is obtained by summing the cost over all the shortest st-paths in G. Similarly, for every $s, t, v \in V$, the value $\sigma_G(s, t, v)$ is obtained by summing the cost over all the shortest st-paths in G that contain v. The betweenness centrality of vertex v is defined as:

$$\frac{1}{\alpha(v)} \sum_{s,t \in V \setminus v} \beta(s)\beta(t) \frac{\sigma_G(s, t, v)}{\sigma_G(s, t)}.$$

Note that if all weights are equal to 1 then this is exactly the definition of Betweenness Centrality for unweighted graphs.

Lemma 10. *Let (A, B) be a split of $G = (V, E, \alpha, \beta)$ and let $C = N_G(B) \subseteq A$, $D = N_G(A) \subseteq B$. Furthermore, let $G_A = (A \cup \{b\}, E_A, \alpha_A, \beta_A)$, $G_B = (B \cup \{a\}, E_B, \alpha_B, \beta_B)$ be the corresponding subgraphs of G, where:*

$$\begin{cases} \alpha_A(v) = \alpha(v), \quad \beta_A(v) = \beta(v) \quad \text{if } v \in A \\ \alpha_B(u) = \alpha(u), \quad \beta_B(u) = \beta(u) \quad \text{if } u \in B \\ \alpha_A(b) = \sum_{u \in D} \alpha(u), \quad \beta_A(b) = \sum_{u \in B} \beta(u) \\ \alpha_B(a) = \sum_{v \in C} \alpha(v), \quad \beta_B(a) = \sum_{v \in A} \beta(v). \end{cases}$$

Then for every $v \in A$ we have:

$$BC_G(v) = BC_{G_A}(v) + [v \in C] BC_{G_B}(a).$$

Proof. Let $v \in A$ be fixed. We consider all possible pairs $s, t \in V \setminus v$ such that $dist_G(s, t) = dist_G(s, v) + dist_G(v, t)$.

Suppose that $s, t \in A \setminus v$. Since (A, B) is a split, the shortest st-paths in G are contained in $N_G[A] = A \cup D$. In particular, the shortest st-paths in G_A are obtained from the shortest st-paths in G by replacing any vertex $d \in D$ by the split marker vertex b. Conversely, the shortest st-paths in G are obtained from the shortest st-paths in G_A by replacing b with any vertex $d \in D$. Hence, $\sigma_{G_A}(s, t, b) = \sum_{d \in D} \sigma_G(s, t, d)$, that implies $\sigma_G(s, t) = \sigma_{G_A}(s, t)$. Furthermore, $\sigma_G(s, t, v) = \sigma_G(s, v)\sigma_G(v, t) = \sigma_{G_A}(s, v)\sigma_{G_A}(v, t) = \sigma_{G_A}(s, t, v)$. As a result, $\sigma_G(s, t, v)/\sigma_G(s, t) = \sigma_{G_A}(s, t, v)/\sigma_{G_A}(s, t)$.

Next, suppose that $s \in B$, $t \in A \setminus v$. Every shortest st-path in G is the concatenation of a shortest sD-path with a shortest tC-path. Therefore, $\sigma_G(s, t) = \frac{\sigma_{G_B}(s,a) \cdot \sigma_{G_A}(b,t)}{\alpha_B(a) \cdot \alpha_A(b)}$. We can furthermore observe v is on a shortest st-path in G if, and only if, v is on a shortest bt-path in G_A. Then, $\sigma_G(s, t, v) = \sigma_G(s, v)\sigma_G(v, t) = \frac{\sigma_{G_B}(s,a) \cdot \sigma_{G_A}(b,v)}{\alpha_B(a) \cdot \alpha_A(b)} \sigma_{G_A}(v, t)$. As a result, $\sigma_G(s, t, v)/\sigma_G(s, t) = \sigma_{G_A}(b, t, v)/\sigma_{G_A}(b, t)$.

Finally, suppose that $s, t \in B$. Again, since (A, B) is a split the shortest st-paths in G are contained in $N_G[B] = B \cup C$. In particular, $\sigma_G(s, t, v) \neq 0$ if, and only if, we have $v \in C$ and $\sigma_{G_B}(s, t, a) \neq 0$. More generally, if $v \in C$ then $\sigma_G(s, t, v) = \frac{\alpha_A(v)}{\alpha_B(a)} \sigma_{G_B}(s, t, a)$. As a result, if $v \in C$ then $\sigma_G(s, t, v)/\sigma_G(s, t) = \frac{\alpha_A(v)}{\alpha_B(a)} \cdot \sigma_{G_B}(s, t, a)/\sigma_{G_B}(s, t)$.

Overall, we have:

$$
\begin{aligned}
BC_G(v) &= \frac{1}{\alpha(v)} \sum_{s,t \in V \setminus v} \beta(s)\beta(t) \frac{\sigma_G(s,t,v)}{\sigma_G(s,t)} \\
&= \frac{1}{\alpha(v)} \sum_{s,t \in A \setminus v} \beta(s)\beta(t) \frac{\sigma_G(s,t,v)}{\sigma_G(s,t)} + \frac{1}{\alpha(v)} \sum_{s \in B,\ t \in A \setminus v} \beta(s)\beta(t) \frac{\sigma_G(s,t,v)}{\sigma_G(s,t)} \\
&\quad + \frac{1}{\alpha(v)} \sum_{s,t \in B} \beta(s)\beta(t) \frac{\sigma_G(s,t,v)}{\sigma_G(s,t)} \\
&= \frac{1}{\alpha_A(v)} \sum_{s,t \in A \setminus v} \beta_A(s)\beta_A(t) \frac{\sigma_{G_A}(s,t,v)}{\sigma_{G_A}(s,t)} + \frac{1}{\alpha_A(v)} \sum_{s \in B,\ t \in A \setminus v} \beta_B(s)\beta_A(t) \frac{\sigma_{G_A}(b,t,v)}{\sigma_{G_A}(b,t)} \\
&\quad + \frac{1}{\alpha_A(v)} [v \in C] \sum_{s,t \in B} \beta_B(s)\beta_B(t) \frac{\alpha_A(v)}{\alpha_B(a)} \cdot \frac{\sigma_{G_B}(s,t,a)}{\sigma_{G_B}(s,t)} \\
&= \left(BC_{G_A}(v) - \frac{\beta_A(b)}{\alpha_A(v)} \sum_{t \in A \setminus v} \beta_A(t) \frac{\sigma_{G_A}(b,t,v)}{\sigma_{G_A}(b,t)} \right) + \frac{\sum_{s \in B} \beta(s)}{\alpha_A(v)} \sum_{t \in A \setminus v} \beta_A(t) \frac{\sigma_{G_A}(b,t,v)}{\sigma_{G_A}(b,t)} \\
&\quad + [v \in C] BC_{G_B}(a) \\
&= BC_{G_A}(v) + [v \in C] BC_{G_B}(a),
\end{aligned}
$$

that finally proves the lemma. □

Theorem 12. *For every* $G = (V, E)$, Betweenness Centrality *can be solved in* $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n + m)$*-time.*

Proof. Let T be a split decomposition tree of G, with its nodes being in bijective correspondance with the split components $C_1, C_2, \ldots, C_k$. It can be computed in linear time [22]. As for Theorem 9, we root T in C_1. For every $1 \leq i \leq k$, let T_i be the subtree of T that is rooted in C_i. If $i > 1$ then let $C_{p(i)}$ be its parent in T. We recall that by construction of T, the edge $\{C_{p(i)}, C_i\} \in E(T)$ corresponds to a split (A_i, B_i) of G. Furthermore, let $G_{A_i} = (A_i \cup \{b_i\}, E_{A_i})$, $G_{B_i} = (B_i \cup \{a_i\}, E_{B_i})$ be the corresponding subgraphs of G. W.l.o.g. we have $C_i \subseteq G_{A_i}$. We observe that T_i is a split decomposition tree of G_{A_i}, while $T \setminus T_i$ is a split decomposition tree of G_{B_i}.

Let us assume $G = (V, E, \alpha, \beta)$ to be vertex-weighted, with initially $\alpha(v) = \beta(v) = 1$ for every $v \in V$. For every $i > 1$, let $G_{A_i} = (A_i \cup \{b_i\}, E_{A_i}, \alpha_{A_i}, \beta_{A_i})$, $G_{B_i} = (B_i \cup \{a_i\}, E_{B_i}, \alpha_{B_i}, \beta_{B_i})$ be as described in Lemma 10. In particular, for every $i > 1$:

$$
\begin{cases}
\alpha_{A_i}(v) = \alpha(v) = 1, \quad \beta_{A_i}(v) = \beta(v) = 1 \quad \text{if } v \in A_i \\
\alpha_{B_i}(u) = \alpha(u) = 1, \quad \beta_{B_i}(u) = \beta(u) = 1 \quad \text{if } u \in B_i \\
\alpha_{A_i}(b_i) = |N_G(A_i)|, \quad \beta_{A_i}(b_i) = |B_i| \\
\alpha_{B_i}(a_i) = |N_G(B_i)|, \quad \beta_{B_i}(a_i) = |A_i|.
\end{cases}
$$

Hence, all the weights can be computed in linear time by dynamic programming over T. We set $G_1 = G$ while $G_i = G_{A_i}$ for every $i > 1$. Furthermore, we first aim at computing $BC_{G_i}(v)$ for every $v \in V(C_i)$.

If C_i is a leaf of T then there are three cases to be considered.

1. Suppose G_i is a complete graph. Then, for every $v \in V(C_i)$ we have $BC_{G_i}(v) = 0$.

2. Suppose G_i is a star, with center node r. In particular, $BC_{G_i}(v) = 0$ for every $v \in V(C_i) \setminus \{r\}$. Furthermore, since r is onto the unique shortest path between every two leaves $s, t \in V(C_i) \setminus \{r\}$, we have $\sigma_{G_i}(s,t,r) = \sigma_{G_i}(s,t)$. Let us write $\beta(G_i) = \sum_{v \in V(C_i)\setminus\{r\}} \beta_{G_i}(v)$. We have:

$$\begin{aligned} BC_{G_i}(r) &= \frac{1}{\alpha_{G_i}(r)} \sum_{s,t \in V(C_i)\setminus\{r\}} \beta_{G_i}(s)\beta_{G_i}(t) \\ &= \frac{1}{2\alpha_{G_i}(r)} \sum_{s \in V(C_i)\setminus\{r\}} \beta_{G_i}(s) \left(\sum_{t \in V(C_i)\setminus\{r,s\}} \beta_{G_i}(t) \right) \\ &= \frac{1}{2\alpha_{G_i}(r)} \sum_{s \in V(C_i)\setminus\{r\}} \beta_{G_i}(s) \left(\beta(G_i) - \beta_{G_i}(s)\right). \end{aligned}$$

   It can be computed in $\mathcal{O}(|V(C_i)|)$-time.

3. Finally, suppose G_i is prime for split decomposition. Brandes algorithm [19] can be generalized to that case. For every $v \in V(C_i)$, we first compute a BFS ordering from v. It takes $\mathcal{O}(|E(C_i)|)$-time. Furthermore for every $u \in V(C_i) \setminus \{v\}$, let $N^+(u)$ be the neighbours $w \in N_{C_i}(u)$ such that w is on a shortest uv-path. We compute $\sigma_{G_i}(u,v)$ by dynamic programming. Precisely, $\sigma_{G_i}(v,v) = \alpha_{G_i}(v)$, and for every $u \neq v$, $\sigma_{G_i}(u,v) = \alpha_{G_i}(u) \cdot \left(\sum_{w \in N^+(u)} \sigma_{G_i}(w,v)\right)$. It takes $\mathcal{O}(|E(C_i)|)$-time.

   Overall in $\mathcal{O}(|V(C_i)||E(C_i)|)$-time, we have computed $\sigma_{G_i}(u,v)$ and $dist_{G_i}(u,v)$ for every $u, v \in V(C_i)$. Then, for every $v \in V(C_i)$, we can compute $BC_{G_i}(v)$ in $\mathcal{O}(|V(C_i)|^2)$-time by enumerating all the pairs $s, t \in V(C_i) \setminus \{v\}$. Since G_i is prime, the total running time is in $\mathcal{O}(|V(C_i)|^3)$, and so, since $\mathcal{O}(|V(C_i)|) = \mathcal{O}(\mathsf{sw}(G))$, in $\mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_i)|)$.

Otherwise, C_i is an internal node of T. Let $C_{i_1}, C_{i_2}, \ldots, C_{i_k}$ be the children of C_i in T. Assume that, for every $1 \leq t \leq k$, $BC_{G_{i_t}}(b_{i_t})$ has been computed (by dynamic programming over T). Let us define the following weight functions for C_i:

$$\begin{cases} \alpha_i(a_{i_t}) = \alpha_{B_{i_t}}(a_{i_t}), \quad \beta_i(a_{i_t}) = \beta_{B_{i_t}}(a_{i_t}) \\ \alpha_i(v) = \alpha_{A_i}(v), \quad \beta_i(v) = \beta_{A_i}(v) \quad \text{otherwise.} \end{cases}$$

Observe that every edge $\{C_i, C_{i_t}\}$ also corresponds to a split (A'_{i_t}, B'_{i_t}) of G_i, where $C_{i_t} \subseteq G_{A'_{i_t}} = G_{A_{i_t}}$. By applying all the corresponding simple decompositions, one finally obtains the subgraph $H_i = (V(C_i), E(C_i), \alpha_i, \beta_i)$. Then, let us define $\ell_i(a_{i_t}) = BC_{G_{i_t}}(b_{i_t})$ and $\ell_i(v) = 0$ else. Intuitively, the function ℓ_i is a corrective term updated after each simple decomposition. More precisely, we obtain by multiple applications of Lemma 10, for every $v \in V(C_i)$:

$$BC_{G_i}(v) = BC_{H_i}(v) + \sum_{u \in N_{H_i}(v)} \ell_i(u)$$

Clearly, this can be reduced in $\mathcal{O}(|E(H_i)|)$-time, resp. in $\mathcal{O}(|V(H_i)|)$-time when H_i is complete, to the computation of $BC_{H_i}(v)$. So, it can be done in $\mathcal{O}(\mathsf{sw}(G)^2 \cdot |V(C_i)|)$-time (*i.e.*, as explained for the case of leaf nodes).

Overall, this first part of the algorithm takes time $\mathcal{O}(\mathsf{sw}(G)^2 \cdot \sum_i |V(C_i)|) = \mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$. Furthermore, since $G_1 = G$, we have computed $BC_G(v)$ for every $v \in V(C_1)$. Then, using the same techniques as above, we can compute $BC_{G_{B_i}}(a_i)$ for every $i > 1$ by reverse dynamic programming over T. It takes $\mathcal{O}(\mathsf{sw}(G)^2 \cdot n)$-time. Finally, by Lemma 10 we can compute $BC_G(v)$ from $BC_{G_i}(v)$ and $BC_{G_{B_i}}(a_i)$, for every $v \in V(C_i)$. It takes linear time. □

4.4 Kernelization methods with modular decomposition

The purpose of the subsection is to show how to apply the previous results, obtained with split decomposition, to modular decomposition. On the way, improvements are obtained for the running time. Indeed, it is often the case that only the quotient graph G' needs to be considered. Then, we derive from modular decomposition a simple *Kernelization* algorithm whose output (called kernel) is G'. We thus obtain algorithms that run in $\mathcal{O}(\mathsf{mw}(G)^{\mathcal{O}(1)} + n + m)$-time. See [84] for an extended discussion on the use of kernelization for graph problems in P.

We start with the following lemma, that we prove for completeness:

Lemma 11 (folklore)**.** *For every $G = (V, E)$ we have $\mathsf{sw}(G) \leq \mathsf{mw}(G) + 1$.*

Proof. First we claim that $\mathsf{mw}(H) \leq \mathsf{mw}(G)$ for every *induced* subgraph H of G. Indeed, for every module M of G we have that $M \cap V(H)$ is a module of H, thereby proving the claim. We show in what follows that a "split decomposition" can be computed from the modular decomposition of G such that all the non degenerate split components have size at most $\mathsf{mw}(G) + 1$[6]. Applying this result to every prime split component of G in its canonical split decomposition proves the lemma.

W.l.o.g., G is connected (otherwise, we consider each connected component separately). Let $\mathcal{M}(G) = \{M_1, M_2, \ldots, M_k\}$ ordered by decreasing size.

1. If $|M_1| = 1$ (G is either complete or prime for modular decomposition) then we output G.
2. Otherwise, suppose $|M_1| < n-1$. We consider all the maximal strong modules $M_1, M_2, \ldots, M_t$ such that $|M_i| \geq 2$ sequentially. For every $1 \leq i \leq t$, we have that $(M_i, V \setminus M_i)$ is a split. Furthermore if we apply the corresponding simple decomposition then we obtain two subgraphs, one being the subgraph G_i obtained from $G[M_i]$ by adding a universal vertex b_i, and the other being obtained from G by replacing M_i by a unique vertex a_i with neighbourhood $N_G(M_i)$. Then, there are two subcases.
   - Subcase $\mathcal{M}(G) = \{M_1, M_2\}$. In particular, $|M_2| \geq 2$. We perform a simple decomposition for M_1. The two resulting subgraphs are exactly G_1 and G_2.
   - Subcase $\{M_1, M_2\} \subsetneq \mathcal{M}(G)$. We apply simple decompositions for $M_1, M_2, \ldots, M_t$ sequentially. Indeed, let $i \in \{1, \ldots, t\}$ and suppose we have already applied simple decompositions for $M_1, M_2, \ldots, M_{i-1}$. Then, since there are at least three modules in $\mathcal{M}(G)$ we have that $(M_i, \{a_1, a_2, \ldots a_{i-1}\} \cup \bigcup_{j>i} M_j)$ remains a split, and so, we can apply a simple decomposition. The resulting components are exactly: the quotient graph G' and, for every $1 \leq i \leq t$, the subgraph G_i obtained from $G[M_i]$.

   Furthermore, in both subcases we claim that the modular decomposition of G_i can be updated from the modular decomposition of $G[M_i]$ in constant time. Indeed, the set of all universal

[6]Formally this is only a partial split decomposition, since there are subgraphs that could be further decomposed.

vertices in a graph is a clique and a maximal strong module. We output G' (only if $\{M_1, M_2\} \subsetneq \mathcal{M}(G)$) and, for every $1 \leq i \leq t$, we apply the procedure recursively for G_i.

3. Finally, suppose $|M_1| = n - 1$. In particular, $\mathcal{M}(G) = \{M_1, M_2\}$ and M_2 is trivial. Let $\mathcal{M}(G[M_1]) = \{M'_1, M'_2, \ldots, M'_p\}$ ordered by decreasing size. If $|M'_1| = 1$ (*i.e.*, $G[M_1]$ is either edgeless, complete or prime for modular decomposition) then we output G. Otherwise we apply the previous Step 2 to the modular partition $M'_1, M'_2, \ldots, M'_p, M_2$.

The procedure takes linear time if the modular decomposition of G is given. Furthermore, the subgraphs obtained are either: the quotient graph G'; a prime subgraph for modular decomposition with an additional universal vertex; or a degenerate graph (that is obtained from either a complete subgraph or an edgeless subgraph by adding a universal vertex). □

Corollary 13. *For every graph G we can solve:*

- Eccentricities *and* Diameter *in $\mathcal{O}(\mathsf{mw}(G)^2 \cdot n + m)$-time;*
- Hyperbolicity *in $\mathcal{O}(\mathsf{mw}(G)^3 \cdot n + m)$-time;*
- Betweenness Centrality *in $\mathcal{O}(\mathsf{mw}(G)^2 \cdot n + m)$-time.*

In what follows, we explain how to improve the above running times in some cases.

Theorem 14. *For every $G = (V, E)$,* Eccentricities *can be solved in $\mathcal{O}(\mathsf{mw}(G)^3 + n + m)$-time. In particular,* Diameter *can be solved in $\mathcal{O}(\mathsf{mw}(G)^3 + n + m)$-time.*

Proof. W.l.o.g., G is connected. Consider the (partial) split decomposition obtained from the modular decomposition of G (Lemma 11). Let T be the corresponding split decomposition tree. By construction, there exists a modular partition $M_1, M_2, \ldots, M_k$ of G with the two following properties:

- All but at most one split components of G are split components of some G_i, $1 \leq i \leq k$, where the graph G_i is obtained from $G[M_i]$ by adding a universal vertex b_i.
- Furthermore, the only remaining split component (if any) is the graph G' obtained by replacing every module M_i with a single vertex a_i. Either G' is degenerate (and so, $diam(G') \leq 2$) or $k \leq \mathsf{mw}(G) + 1$. We can also observe in this situation that if we root T in G' then the subtrees of $T \setminus \{G'\}$ are split decomposition trees of the graphs G_i, $1 \leq i \leq k$.

We can solve Eccentricities for G as follows. First for every $1 \leq i \leq k$ we solve Eccentricities for G_i. In particular, $diam(G_i) \leq 2$, and so, for every $v \in V(G_i)$ we have: $ecc_{G_i}(v) = 0$ if and only if $V(G_i) = \{v\}$; $ecc_{G_i}(v) = 1$ if and only if v is universal in G_i; otherwise, $ecc_{G_i}(v) = 2$. Therefore, we can solve Eccentricities for G_i in $\mathcal{O}(|V(G_i)| + |E(G_i)|)$-time. Overall, this step takes $\mathcal{O}(\sum_{i=1}^{k} |V_i| + |E_i|) = \mathcal{O}(n + m)$-time. Then there are two subcases.

Suppose G' is not a split component. We deduce from Lemma 11 $G = G[M_1] \oplus G[M_2]$, *i.e.*, G is obtained from $G[M_1]$ and $G[M_2]$ by adding all possible edges between M_1 and M_2. In this situation, for every $i \in \{1, 2\}$, for every $v \in V(G_i)$ we have $ecc_G(v) = \max\{ecc_{G_i}(v), 1\}$.

Otherwise, let us compute Eccentrities for G'. It takes $\mathcal{O}(|V(G')|) = \mathcal{O}(n)$-time if G' is degenerate, and $\mathcal{O}(\mathsf{mw}(G)^3)$-time otherwise. Applying the algorithmic scheme of Theorem 9, one obtains $ecc_G(v) = \max\{ecc_{G_i}(v), dist_{G_i}(v, a_i) + ecc_{G'}(b_i) - 1\} = \max\{ecc_{G_i}(v), ecc_{G'}(b_i)\}$ for every $v \in M_i$. Hence, we can compute $ecc_G(v)$ for every $v \in V$ in $\mathcal{O}(n)$-time. □

Corollary 15. *For every connected graph G, $diam(G) \leq \max\{2, \mathsf{mw}(G) - 1\}$.*

Proof. By the proof of the above Theorem 14, we have for every G with quotient graph G' that $diam(G) \leq \max\{2, diam(G')\}$. Since $diam(G') \leq |V(G')| - 1 \leq \mathsf{mw}(G) - 1$, the result follows directly. □

Next, we consider Hyperbolicity. It is proved in [96] that, for every G with quotient graph G', $\delta(G') \leq \delta(G) \leq \max\{\delta(G'), 1\}$. The latter immediately implies the following result:

Theorem 16. *For every graph G, we can decide whether $\delta(G) > 1$, and if so, compute $\delta(G)$, in $\mathcal{O}(\mathsf{mw}(G)^4 + n + m)$-time.*

However, we did not find a way to preprocess G in linear time so that we can compute $\delta(G)$ from $\delta(G')$. Indeed, let G_M be a graph of diameter at most 2. Solving Eccentricities for G_M can be easily done in linear time. However, the following shows that it is not that simple to do so for Hyperbolicity.

Lemma 12 ([28]). *For every graph G we have $\delta(G) \leq \lfloor diam(G)/2 \rfloor$. Furthermore, if $diam(G) \leq 2$ then $\delta(G) < 1$ if and only if G is C_4-free.*

The detection of an induced C_4 in $\mathcal{O}(\mathsf{mw}(G)^{\mathcal{O}(1)} + n + m)$-time remains an open problem.

Short digression: using neighbourhood diversity

We show that by imposing more constraints on the modular partition, some more kernels can be computed for the problems in Section 4.1. Two vertices u, v are *twins* in G if $N_G(u) \setminus v = N_G(v) \setminus u$. Being twins induce an equivalence relationship over $V(G)$. The number of equivalence classes is called the *neighbourhood diversity* of G, which we denote by $\mathsf{nd}(G)$ [81]. Observe that every set of pairwise twins is a module of G. Hence, $\mathsf{mw}(G) \leq \mathsf{nd}(G)$.

Theorem 17. *For every $G = (V, E)$,* Hyperbolicity *can be solved in $\mathcal{O}(\mathsf{nd}(G)^4 + n + m)$-time.*

Proof. Let $V_1, V_2, \ldots, V_k$, $k = \mathsf{nd}(G)$, partition the vertex-set V in twin classes. The partition can be computed in linear time [81]. Furthermore, since it is a modular partition, we can compute a (partial) split decomposition as described in Lemma 11. Let $G' = (V', E')$ such that $V' = \{v_1, v_2, \ldots, v_k\}$ and $E' = \{\{v_i, v_j\} \mid V_i \times V_j \subseteq E\}$. Then, the split components are either: G', stars S^i (if the vertices of V_i are pairwise nonadjacent, *i.e.*, false twins) or complete graphs K^i (if the vertices of V_i are pairwise adjacent, *i.e.*, true twins).

Applying the algorithmic scheme of Theorem 10, in order to solve Hyperbolicity for G it suffices to compute, for every split component C_j, the hyperbolicity value $\delta(C_j)$ *and* all the simplicial vertices in C_j. This can be done in $\mathcal{O}(|V(C_j)|)$-time if C_j is a star or a complete graph, and in $\mathcal{O}(\mathsf{nd}(G)^4)$-time if $C_j = G'$. Therefore, we can solve Hyperbolicity for G in total $\mathcal{O}(\mathsf{nd}(G)^4 + n + m)$-time. □

In [49], the authors propose an $\mathcal{O}(2^{\mathcal{O}(k)} + n + m)$-time algorithm for computing Hyperbolicity with k being the vertex-cover number of the graph. Their algorithm is pretty similar to Theorem 17. This is no coincidence since every graph with vertex-cover at most k has neighbourhood diversity at most $2^{\mathcal{O}(k)}$ [81].

Finally, the following was proved implicitly in [89].

Theorem 18 ([89]). *For every graph G,* Betweenness Centrality *can be solved in $\mathcal{O}(\mathsf{nd}(G)^3 + n + m)$-time.*

4.5 Applications to graphs with few P_4's

Before ending Section 4, we apply the results of the previous subsections to the case of $(q, q-3)$-graphs. For that we need to consider all the cases where the quotient graph has super-constant size $\Omega(q)$ (see Lemma 4).

Eccentricities

Theorem 19. *For every graph G,* Eccentricities *can be solved in $\mathcal{O}(\mathsf{q}(G)^3 + n + m)$-time.*

Proof. By Lemma 11, there exists a partial split decomposition of G such that the only split component with diameter possibly larger than 2 is its quotient graph G'. Furthermore, as shown in the proof of Theorem 14, solving Eccentricities for G can be reduced in $\mathcal{O}(n+m)$-time to the solving of Eccentricities for G'. By Lemma 4 we only need to consider the following cases. We can check in which case we are in linear time [6].

- Suppose $G' = (S' \cup K' \cup R', E')$ is a prime spider. There are two subcases.
  1. If G' is a thick spider then it has diameter two. Since in addition, there is no universal vertex in G', therefore every vertex of G' has eccentricity exactly two.
  2. Otherwise, G' is a thin spider. Since there is no universal vertex in G', every vertex has eccentricity at least two. In particular, since K' is a clique dominating set of G', $ecc_{G'}(v) = 2$ for every $v \in K'$. Furthermore, since there is a join between K' and R', $ecc_{G'}(v) = 2$ for any $v \in R'$. Finally, since every two vertices of S' are pairwise at distance three, $ecc_{G'}(v) = 3$ for every $v \in S$.
- Suppose G' is isomorphic either to a cycle $C_{n'}$, or to a co-cycle $\overline{C_{n'}}$, for some $n' \geq 5$.
  1. If G' is isomorphic to a cycle $C_{n'}$ then every vertex of G' has eccentricity $\lfloor n'/2 \rfloor$.
  2. Otherwise, G' is isomorphic to a co-cycle $\overline{C_{n'}}$. We claim that every vertex of G' has eccentricity 2. Indeed, let $v \in \overline{C_{n'}}$ be arbitrary and let $u, w \in \overline{C_{n'}}$ be the only two vertices nonadjacent to v. Furthermore, let u', w' be the unique vertices of $\overline{C_{n'}} \setminus v$ that are respectively nonadjacent to u and to w. Since $n' \geq 5$, we have $u' \neq w'$. In particular, (v, u', w) and (v, w', u) are, respectively, a shortest vw-path and a shortest vu-path. Hence, $ecc_{G'}(v) = 2$.
- Suppose G' is a spiked p-chain P_k, or its complement.
  - Subcase G' is a spiked p-chain P_k. In particular, G' contains the k-node path $P_k = (v_1, v_2, \ldots, v_k)$ as an isometric subgraph. Furthermore, if $x \in V(G')$ then $dist_{G'}(v_1, x) = 2$, and x and v_2 are twins in $G' \setminus v_1$. Similarly, if $y \in V(G')$ then $dist_{G'}(v_k, y) = 2$, and y and v_{k-1} are twins in $G' \setminus v_k$. As a result: for every $1 \leq i \leq k$, $ecc_{G'}(v_i) = ecc_{P_k}(v_i) = \max\{i-1, k-i\}$; if $x \in V(G')$ then $ecc_{G'}(x) = ecc_{P_k}(v_2) = k-2$; if $y \in V(G')$ then $ecc_{G'}(y) = ecc_{P_k}(v_{k-1}) = k-2$.
  - Subcase G' is a spiked p-chain $\overline{P_k}$. In particular, $\overline{G'}$ is a spiked p-chain P_k. Since $k \geq 6$, every spiked p-chain P_k has diameter more than four. Hence, $diam(G') \leq 2$, that implies Eccentricities can be solved for G' in linear time.

- Suppose G' is a spiked p-chain Q_k, or its complement.
  - Subcase G' is a spiked p-chain Q_k. There is a clique-dominating set $K' = \{v_2, v_4, \ldots, v_{2j}, \ldots\}$ of G'. In particular, every vertex of K' has eccentricity two. Furthermore, any z_i is adjacent to both v_2, v_4. Every vertex v_{2i-1}, except v_3, is adjacent to v_2. Finally, every vertex v_{2i-1}, except v_1 and v_5, is adjacent to v_4. As a result, any vertex z_i has eccentricity two; any vertex $v_{2i-1},\ i \notin \{1,2,3\}$, also has eccentricity two. However, since v_2 and v_4 are, respectively, the only neighbours of v_1 and v_3, we get $dist_{G'}(v_1, v_3) = dist_{G'}(v_3, v_5) = 3$. Hence $ecc_{G'}(v_1) = ecc_{G'}(v_3) = ecc_{G'}(v_5) = 3$.
  - Subcase G' is a spiked p-chain $\overline{Q_k}$. Roughly, we reverse the roles of vertices v_{2i} with even index with the roles of vertices v_{2i-1} with odd index. More precisely, there is a clique-dominating set $K' = \{v_1, v_3, \ldots, v_{2j-1}, \ldots\}$ of G'. In particular, every vertex of K' has eccentricity two. Furthermore, any z_i is adjacent to both v_1, v_3. Every vertex v_{2i}, except v_2, is adjacent to v_1. Finally, every vertex v_{2i}, except v_4, is adjacent to v_3. As a result, any vertex z_i has eccentricity two; any vertex $v_{2i},\ i \notin \{1,2\}$, also has eccentricity two. However, since v_3 is the only neighbour of v_2, we get $dist_{G'}(v_2, v_4) = 3$, hence $ecc_{G'}(v_2) = ecc_{G'}(v_4) = 3$.
- Otherwise, $|V(G')| \leq \mathsf{q}(G)$. Then, solving Eccentricities for G' can be done in $\mathcal{O}(\mathsf{q}(G)^3)$-time.

Therefore, in all the above cases, Eccentricities can be solved for G' in $\mathcal{O}(\min\{\mathsf{q}(G)^3, n+m\})$-time. □

Corollary 20. *For every connected $(q, q-4)$-graph G, $diam(G) \leq q$.*

Corollary 20 does not hold for $(q, q-3)$-graphs because of cycles and spiked p-chains P_k.

Gromov hyperbolicity

Theorem 21. *For every graph G,* Hyperbolicity *can be solved in $\mathcal{O}(\mathsf{q}(G)^3 \cdot n + m)$-time.*

Proof. By Lemma 11, we can compute a partial split decomposition from the modular decomposition of G. It takes $\mathcal{O}(n+m)$-time. Let $C_1, C_2, \ldots, C_k$ be the split components. By using the algorithmic scheme of Theorem 10, solving Hyperbolicity can be reduced in $\mathcal{O}(\sum_i |V(C_i)| + |E(C_i)|)$-time to the computation, for every $1 \leq i \leq k$, of the hyperbolicity value $\delta(C_i)$ *and* of all the simplicial vertices in C_i. We claim that it can be done in $\mathcal{O}(\mathsf{q}(G)^3 \cdot |V(C_i)| + |E(C_i)|)$-time. Since $\sum_i |V(C_i)| = \mathcal{O}(n)$ and $\sum_i |E(C_i)| = \mathcal{O}(n+m)$ [92], the latter claim will prove the desired time complexity.

If C_i is degenerate then the above can be done in $\mathcal{O}(|V(C_i)|)$-time. Otherwise, C_i is obtained from a prime subgraph G' in the modular decomposition of G by possibly adding a universal vertex. In particular, we have: $\delta(G') = \delta(C_i)$ if $G' = C_i$; $\delta(C_i) = 0$ can be decided in $\mathcal{O}(|V(C_i)| + |E(C_i)|)$-time [72]; otherwise, $diam(C_i) \leq 2$, and so, by Lemma 12 we have $\delta(C_i) = 1$ if and only if C_i contains an induced cycle of length four (otherwise, $\delta(C_i) = 1/2$). Therefore, we are left to compute the following for every prime subgraph G' in the modular decomposition of G:

- Compute $\delta(G')$;
- Decide whether G' contains an induced cycle of length four;

- Compute the simplicial vertices in G'.

In particular, if $|V(G')| \leq \mathsf{q}(G)$ then it can be done in $\mathcal{O}(|V(G')|^4) = \mathcal{O}(\mathsf{q}(G)^3 \cdot |V(G')|)$-time. Otherwise, by Lemma 4 we only need to consider the following cases. We can check in which case we are in linear time [6].

- Suppose G' is a prime spider. In particular it is a split graph, and so, it does not contain an induced cycle of length more than three. Furthermore the simplicial vertices of any chordal graph, and so, of G', can be computed in linear time. If G' is a thin spider then it is a block-graph, and so, $\delta(G') = 0$ [72]. Otherwise, G' is a thick spider, and so, it contains an induced diamond. The latter implies $\delta(G') \geq 1/2$. Since $diam(G') \leq 2$ and G' is C_4-free, by Lemma 12 $\delta(G') < 1$, hence we have $\delta(G') = 1/2$.

- Suppose G' is a cycle or a co-cycle of order at least five. Since cycles and co-cycles are non complete regular graphs they do not contain any simplicial vertex [4]. Furthermore, a cycle of length at least five of course does not contain an induced cycle of length four; a co-cycle of order five is a C_5, and a co-cycle or order at least six always contains an induced cycle of length at least four since there is an induced $2K_2 = \overline{C_4}$ in its complement. Finally, the hyperbolicity of a given cycle can be computed in linear time [23]; for every co-cycle of order at least six, since it has diameter at most two and it contains an induced cycle of length four, by Lemma 12 it has hyperbolicity equal to 1.

- Suppose G' is a spiked p-chain P_k, or its complement. In particular, if G' is a spiked p-chain P_k then it is a block-graph, and so, a chordal graph. It implies $\delta(G') = 0$ [72], G' does not contain any induced cycle of length four, furthermore all the simplicial vertices of G' can be computed in linear time. Else, G' is a spiked p-chain $\overline{P_k}$. Since, in $\overline{G'}$, every vertex is nonadjacent to at least one edge of P_k, it implies that G' has no simplicial vertex. Furthermore, since P_k, and so, $\overline{G'}$, contains an induced $2K_2$, the graph G' contains an induced cycle of length four. Since $diam(G') = 2$, it implies by Lemma 12 $\delta(G') = 1$.

- Otherwise, G' is a spiked p-chain Q_k, or its complement. In both cases, G' is a split graph, and so, a chordal graph. It implies that G' does not contain an induced cycle of length four, and that all the simplicial vertices of G' can be computed in linear time. Furthermore, we can decide in linear time whether $\delta(G') = 0$ [72]. Otherwise, it directly follows from the characterization in [20] that a necessary condition for a chordal graph to have hyperbolicity at least one is to contain two disjoint pairs of vertices at distance 3. Since there are no such pairs in G', $\delta(G') = 1/2$.

□

The solving of Betweenness Centrality for $(q, q-3)$-graphs is left for future work. We think it is doable with the techniques of Theorem 12. However, this would require to find ad-hoc methods for every graph family in Lemma 4. The main difficulty is that we need to consider weighted variants of these graph families, and the possibility to add a universal vertex.

5 New Parameterized algorithms for Maximum Matching

A matching in a graph is a set of edges with pairwise disjoint end vertices. We consider the problem of computing a matching of maximum size.

Problem 8 (Maximum Matching).

Input: *A graph G.*

Output: *A matching of G with maximum cardinality.*

Maximum Matching can be solved in polynomial time with Edmond's algorithm [45]. A naive implementation of the algorithm runs in $\mathcal{O}(n^4)$ time. Nevertheless, Micali and Vazirani [85] show how to implement Edmond's algorithm in time $\mathcal{O}(m\sqrt{n})$. In [84], Mertzios, Nichterlein and Niedermeier design some new algorithms to solve Maximum Matching, that run in $\mathcal{O}(p^{\mathcal{O}(1)} \cdot (n + m))$-time for various graph parameters p. They also suggest to use Maximum Matching as the "drosophilia" of the study of fully polynomial parameterized algorithms.

In this section, we present $\mathcal{O}(k^4 \cdot n + m)$-time algorithms for solving Maximum Matching, when parameterized by either the modular-width or the P_4-sparseness of the graph. The latter subsumes many algorithms that have been obtained for specific subclasses [55, 101].

5.1 Computing short augmenting paths using modular decomposition

Let $G = (V, E)$ be a graph and $F \subseteq E$ be a matching of G. A vertex is termed matched if it is incident to an edge of F, and unmatched otherwise. An F-augmenting path is a path where the two ends are unmatched, all edges $\{x_{2i}, x_{2i+1}\}$ are in F and all edges $\{x_{2j-1}, x_{2j}\}$ are not in F. We can observe that, given an F-augmenting path $P = (x_1, x_2, \ldots, x_{2k})$, the matching $E(P)\Delta F$ (obtained by replacing the edges $\{x_{2i}, x_{2i+1}\}$ with the edges $\{x_{2j-1}, x_{2j}\}$) has larger size than F.

Theorem 22 (Berge, [12]). *A matching F in a graph G is maximum if and only if there is no F-augmenting path.*

We now sketch our approach. Suppose that, for every module $M_i \in \mathcal{M}(G)$, a maximum matching F_i of $G[M_i]$ has been computed. Then, $F = \bigcup_i F_i$ is a matching of G, but it is not necessarily maximum. Our approach consists in computing short augmenting paths (of length $\mathcal{O}(\mathsf{mw}(G))$) using the quotient graph G', until we obtain a maximum matching. For that, we need to introduce several reduction rules.

The first rule (proved below) consists in removing, from every module M_i, the edges that are not part of its maximum matching F_i.

Lemma 13. *Let M be a module of $G = (V, E)$, let $G[M] = (M, E_M)$ and let $F_M \subseteq E_M$ be a maximum matching of $G[M]$. Then, every maximum matching of $G'_M = (V, (E \setminus E_M) \cup F_M)$ is a maximum matching of G.*

Proof. Let us consider an arbitrary maximum matching of G. We totally order $M = \{v_1, v_2, \ldots, v_l\}$, in such a way that unmatched vertices appear first, and for every edge in the matching F_M the two ends of it are consecutive. Let $S \subseteq M$, $|S| = k$ be the vertices of M that are matched with a vertex of $V \setminus M$. We observe that with the remaining $|M| - k$ vertices of $M \setminus S$, we can only obtain a matching of size at most $\mu_M = \min\{|F_M|, \lfloor(|M| - k)/2\rfloor\}$. Conversely, if $S = \{v_1, v_2, \ldots, v_k\}$ then we can always create a matching of size exactly μ_M with the vertices of $M \setminus S$ and the edges of F_M. Since M is a module of G, this choice can always be made without any loss of generality. $\square$

From now on we shall assume each module induces a matching. In particular, for every $M \in \mathcal{M}(G)$, the set $V(E(G[M]))$ stands for the non isolated vertices in the subgraph $G[M]$.

Then, we need to upper bound the number of edges in an augmenting path that are incident to a same module.

Lemma 14. *Let $G = (V, E)$ be a graph such that every module $M \in \mathcal{M}(G)$ induces a matching. Furthermore let $G' = (\mathcal{M}(G), E')$ be the quotient graph of G, and let $F \subseteq E$ be a non maximum matching of G. There exists an F-augmenting path $P = (x_1, x_2, \ldots, x_{2\ell})$ such that the following hold for every $M \in \mathcal{M}(G)$:*

- $|\{i \mid x_{2i-1}, x_{2i} \in M\}| \leq 1$;

  furthermore if $|\{i \mid x_{2i-1}, x_{2i} \in M\}| = 1$ then we have $\{i \mid x_{2i-1}, x_{2i} \in M'\} = \emptyset$ for every $M' \in N_{G'}(M)$;

- $|\{i \mid x_{2i}, x_{2i+1} \in M\}| \leq 1$;

- $|\{i \mid x_{2i-1} \notin M,\ x_{2i} \in M\}| \leq 1$;

- $|\{i \mid x_{2i} \notin M,\ x_{2i+1} \in M\}| \leq 2$;

  furthermore if $|\{i \mid x_{2i} \notin M,\ x_{2i+1} \in M\}| = 2$ then there exist $x_{2i_0+1}, x_{2i_0+3}, x_{2i_0+4} \in M$;

- $|\{i \mid x_{2i-1} \in M,\ x_{2i} \notin M\}| \leq 1$;

- $|\{i \mid x_{2i} \in M,\ x_{2i+1} \notin M\}| \leq 2$;

  furthermore if $|\{i \mid x_{2i} \in M,\ x_{2i+1} \notin M\}| = 2$ then there exist $x_{2i_0-1}, x_{2i_0}, x_{2i_0+2} \in M$.

In particular, P has length $\mathcal{O}(|\mathcal{M}(G)|)$.

Proof. Let P be a *shortest* F-augmenting path that minimizes $i(P) = \left|E(P) \cap \left(\bigcup_{M \in \mathcal{M}(G)} E(G[M])\right)\right|$. Equivalently, P is a shortest augmenting path with the minimum number of edges $i(P)$ with their two ends in a same module. There are four cases.

1. Suppose by contradiction there exist $i_1 < i_2$ such that $x_{2i_1-1}, x_{2i_1}, x_{2i_2-1}, x_{2i_2} \in M$. See Fig. 9-10. In particular, $i_2 - i_1 \geq 2$ since M induces a matching. Furthermore, $x_{2i_1+1}, x_{2i_2-2} \in N_G(M)$. Then, $(x_1, \ldots, x_{2i_1-1}, x_{2i_1+1}, x_{2i_1}, x_{2i_2-2}, x_{2i_2-1}, x_{2i_2}, \ldots x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $i(P)$.

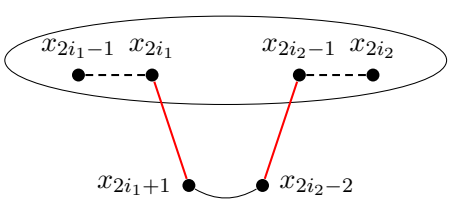

Figure 9: Case $x_{2i_1-1}, x_{2i_1}, x_{2i_2-1}, x_{2i_2} \in M$.

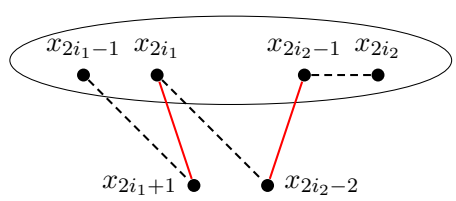

Figure 10: Local replacement of P.

Similarly, suppose by contradiction there exist $x_{2i_1-1}, x_{2i_1} \in M$ and there exist $x_{2i_2-1}, x_{2i_2} \in M'$, $M' \in N_{G'}(M)$. See Fig 11. We assume by symmetry $i_1 < i_2$. In this situation, either $i_1 = 1$, and so, $x_{2i_1-1} = x_1$ is unmatched, or $i_1 > 1$ and so, x_{2i_1-1} is matched to $x_{2i_1-2} \neq x_{2i_2}$. Then, $(x_1, \ldots, x_{2i_1-1}, x_{2i_2}, \ldots, x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $|V(P)|$.

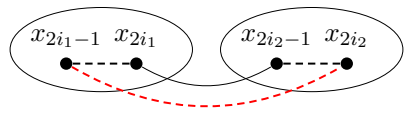

Figure 11: Case $x_{2i_1-1}, x_{2i_1} \in M$ and $x_{2i_2-1}, x_{2i_2} \in M'$.

2. Suppose by contradiction there exist $i_1 < i_2$ such that $x_{2i_1}, x_{2i_1+1}, x_{2i_2}, x_{2i_2+1} \in M$. See Fig 12. In particular, $x_{2i_1-1} \in N_G(M)$ since M induces a matching. Then, $(x_1, \ldots, x_{2i_1-1}, x_{2i_2}, x_{2i_2+1}, \ldots x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $|V(P)|$.

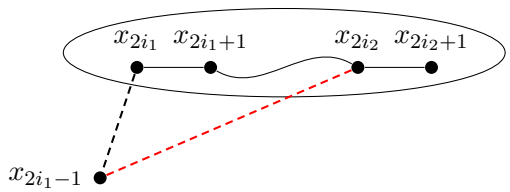

Figure 12: Case $x_{2i_1}, x_{2i_1+1}, x_{2i_2}, x_{2i_2+1} \in M$.

3. Suppose by contradiction there exist $i_1 < i_2$ such that $x_{2i_1-1}, x_{2i_2-1} \notin M,\ x_{2i_1}, x_{2i_2} \in M$. See Fig 13. Either $i_1 = 1$, and so, $x_{2i_1-1} = x_1$ is unmatched, or $i_1 > 1$ and so, x_{2i_1-1} is matched to $x_{2i_1-2} \neq x_{2i_2}$. Then, $(x_1, \ldots, x_{2i_1-1}, x_{2i_2}, \ldots, x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $|V(P)|$.

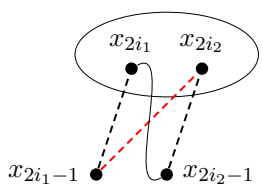

Figure 13: Case $x_{2i_1-1}, x_{2i_2-1} \notin M,\ x_{2i_1}, x_{2i_2} \in M$.

By symmetry, the latter also proves that $|\{i \mid x_{2i-1} \in M,\ x_{2i} \notin M\}| \leq 1$.

4. Finally suppose by contradiction there exist $i_1 < i_2 < i_3$ such that $x_{2i_1}, x_{2i_2}, x_{2i_3} \notin M$, $x_{2i_1+1}, x_{2i_2+1}, x_{2i_3+1} \in M$. See Fig 14. Then, $(x_1, \ldots, x_{2i_1}, x_{2i_1+1}, x_{2i_3}, x_{2i_3+1} \ldots, x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $|V(P)|$.

   We prove in the same way that if there exist $i_1 < i_2$ such that $x_{2i_1}, x_{2i_2} \notin M$ and $x_{2i_1+1}, x_{2i_2+1} \in M$ then $i_2 = i_1 + 1$. See Fig 15. Furthermore, if $x_{2i_2+2} = x_{2i_1+4} \notin M$, then $(x_1, \ldots, x_{2i_1}, x_{2i_1+1}, x_{2i_2+2}, x_{2i_2+3}, \ldots, x_{2\ell})$ is an F-augmenting path, thereby contradicting the minimality of $|V(P)|$.

   By symmetry, the same proof as above applies to $\{i \mid x_{2i} \in M,\ x_{2i+1} \notin M\}$.

Overall, every $M \in \mathcal{M}(G)$ is incident to at most 8 edges of P, and so, P has length $\mathcal{O}(|\mathcal{M}(G)|)$. $\square$

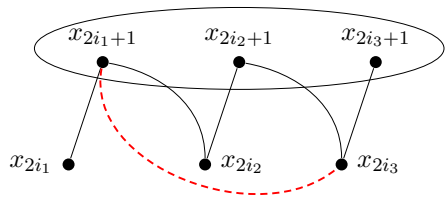

Figure 14: Case $x_{2i_1}, x_{2i_2}, x_{2i_3} \notin M$, $x_{2i_1+1}, x_{2i_2+1}, x_{2i_3+1} \in M$.

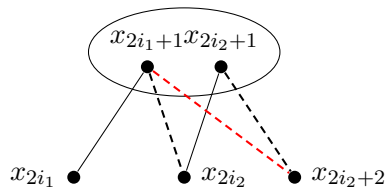

Figure 15: Case $x_{2i_2+2} = x_{2i_1+4} \notin M$.

Based on Lemmas 13 and 14, we introduce in what follows a *witness subgraph* in order to find a matching. We think the construction could be improved but we chose to keep it as simple as possible.

Definition 1. Let $G = (V, E)$ be a graph, $G' = (\mathcal{M}(G), E')$ be its quotient graph and $F \subseteq E$ be a matching of G.

The witness matching F' is obtained from F by keeping a representative for every possible type of edge in an augmenting path. Precisely:

- Let $M \in \mathcal{M}(G)$. If $E(G[M]) \cap F \neq \emptyset$ then there is exactly one edge $\{u_M, v_M\} \in E(G[M]) \cap F$ such that $\{u_M, v_M\} \in F'$. Furthermore if $E(G[M]) \setminus F \neq \emptyset$ then we pick an edge $\{x_M, y_M\} \in E(G[M]) \setminus F$ and we add in F' every edge in F that is incident to either x_M or y_M.
- Let $M, M' \in \mathcal{M}(G)$ be adjacent in G'. There are exactly $\min\{4, |F \cap (M \times M')|\}$ edges $\{v_M, v_{M'}\}$ added in F' such that $v_M \in M$, $v_{M'} \in M'$ and $\{v_M, v_{M'}\} \in F$.

The witness subgraph G'_F is the subgraph induced by $V(F')$ with at most two unmatched vertices added for every strong module. Formally, let $M \in \mathcal{M}(G)$. The submodule $M_F \subseteq M$ contains exactly $\min\{2, |M \setminus V(F)|\}$ vertices of $M \setminus V(F)$. Then,

$$G'_F = G\left[V(F') \cup \left(\bigcup_{M \in \mathcal{M}(G)} M_F\right)\right].$$

As an example, suppose that every edge of F has its two ends in a same module and every module induces a matching. Then, G'_F is obtained from G' by substituting every $M \in \mathcal{M}(G)$ with at most one edge (if $F \cap E(G[M]) \neq \emptyset$) and at most two isolated vertices (representing unmatched vertices).

From the algorithmic point of view, we need to upper bound the size of the witness subgraph, as follows.

Lemma 15. *Let $G = (V, E)$ be a graph, $G' = (\mathcal{M}(G), E')$ be its quotient graph and $F \subseteq E$ be a matching of G. The witness subgraph G'_F has order $\mathcal{O}(|E(G')|)$.*

Proof. By construction for every $M \in \mathcal{M}(G)$ we have $|M \cap V(G'_F)| = \mathcal{O}(deg_{G'}(M))$. Therefore, $|V(G'_F)| = \sum_{M \in \mathcal{M}(G)} |M \cap V(G'_F)| = \mathcal{O}(|E(G')|)$. □

Our algorithm is based on the correspondance between F-augmenting paths in G and F'-augmenting paths in G'_F, that we prove next. The following Lemma 16 is the key technical step of the algorithm.

Lemma 16. *Let $G = (V, E)$ be a graph such that every module $M \in \mathcal{M}(G)$ induces a matching. Let $F \subseteq E$ be a matching of G such that $\bigcup_{M \in \mathcal{M}(G)} V(E(G[M])) \subseteq V(F)$. There exists an F-augmenting path in G if and only if there exists an F'-augmenting path in G'_F.*

Proof. In one direction, G'_F is an induced subgraph of G. Furthermore, according to Definition 1, $F' \subseteq F$ and $V(F') = V(F) \cap V(G'_F)$. Thus, every F'-augmenting path in G'_F is also an F-augmenting path in G.

Conversely, suppose there exists an F-augmenting path in G. Let $P = (v_1, v_2, \ldots, v_{2\ell})$ be an F-augmenting path in G that satisfies the conditions of Lemma 14. We transform P into an F'-augmenting path in G'_F as follows. For every $1 \leq i \leq 2\ell$ let $M_i \in \mathcal{M}(G)$ such that $v_i \in M_i$.

- We choose $u_1 \in M_1 \cap V(G'_F),\ u_{2\ell} \in M_{2\ell} \cap V(G'_F)$ unmatched. Furthermore, if $M_1 = M_{2\ell}$ then we choose $u_1 \neq u_{2\ell}$. The two of $u_1, u_{2\ell}$ exist according to Definition 1.
- Then, for every $1 \leq i \leq \ell - 1$, we choose $u_{2i} \in M_{2i} \cap V(G'_F),\ u_{2i+1} \in M_{2i+1} \cap V(G'_F)$ such that $\{u_{2i}, u_{2i+1}\} \in F'$. Note that if $M_{2i} = M_{2i+1}$ then $\{u_{2i}, u_{2i+1}\}$ is the unique edge of $F' \cap E(G[M_{2i}])$. By Lemma 14 we also have that $\{v_{2i}, v_{2i+1}\}$ is the unique edge of $E(P) \cap F$ such that $v_{2i}, v_{2i+1} \in M_{2i}$. Otherwise, $M_{2i} \neq M_{2i+1}$. If there are p edges $e \in F$ with one end in M_{2i} and the other end in M_{2i+1} then there are at least $\min\{p, 4\}$ such edges in F'. By Lemma 14 there are at most $\min\{p, 4\}$ edges $e \in E(P) \cap F$ with one end in M_{2i} and the other end in M_{2i+1}. Hence, we can always ensure the u_j's, $1 \leq j \leq 2\ell$, to be pairwise different.

The resulting sequence $\mathcal{S}_P = (u_1, u_2, \ldots, u_{2\ell})$ is not necessarily a path, since two consecutive vertices u_{2i-1}, u_{2i} need not be adjacent in G'_F. Roughly, we insert alternating subpaths in the sequence in order to make it a path. However, we have to be careful not to use twice a same vertex for otherwise we would only obtain a walk.

Let $I_P = \{i \mid \{u_{2i-1}, u_{2i}\} \notin E\}$. Observe that for every $i \in I_P$ we have $M_{2i-1} = M_{2i}$. In particular, since we assume $\bigcup_{M \in \mathcal{M}(G)} V(E(G[M])) \subseteq V(F)$, it implies $i \notin \{1, \ell\}$. Furthermore, $M_{2i-2} \neq M_{2i}$ and $M_{2i} \neq M_{2i+1}$ since otherwise $v_{2i-2}, v_{2i-1}, v_{2i} \in M_{2i}$ or $v_{2i-1}, v_{2i}, v_{2i+1} \in M_{2i}$ thereby contradicting that M_{2i} induces a matching. According to Definition 1 there exist $x_i, y_i \in M_{2i}$ such that $\{x_i, y_i\} \in E(G[M_{2i}]) \setminus F$ and every edge of F that is incident to either x_i or y_i is in F'. Such two edges always exist since we assume $\bigcup_{M \in \mathcal{M}(G)} V(E(G[M])) \subseteq V(F)$, hence there exist w_i, z_i such that $\{w_i, x_i\}, \{y_i, z_i\} \in F'$. Note that $w_i, z_i \notin M_{2i}$ since $\{x_i, y_i\} \in E(G[M_{2i}])$ and M_{2i} induces a matching.

Since by Lemma 14 $\{v_{2i-1}, v_{2i}\}$ is the unique edge $e \in E(P) \setminus F$ such that $e \subseteq M_{2i}$, the vertices $x_i, y_i,\ i \in I_P$ are pairwise different. Furthermore, we claim that there can be no $i_1, i_2 \in I_P$ such that $s_{i_1} \in \{x_{i_1}, y_{i_1}\}$ and $t_{i_2} \in \{x_{i_2}, y_{i_2}\}$ are adjacent in G. Indeed otherwise, $M_{2i_1} \in N_{G'}(M_{2i_2})$, there exist $v_{2i_1-1}, v_{2i_1} \in M_{2i_1}$ and $v_{2i_2-1}, v_{2i_2} \in M_{2i_2}$, thereby contradicting Lemma 14. As a result, all

the vertices $w_i, x_i, y_i, z_i,\ i \in I_P$ are pairwise different. However, we may have $\{w_i, x_i\} = \{u_{2j}, u_{2j+1}\}$ or $\{y_i, z_i\} = \{u_{2j}, u_{2j+1}\}$ for some j.

We consider the indices $i \in I_P$ sequentially, by increasing value. By Lemma 14, $v_{2j} \notin M_{2i},\ v_{2j+1} \in M_{2i}$ for some $j \neq i-1$ implies $j = i-2$. Similarly (obtained by reverting the indices, from $v'_1 = v_{2\ell}$ to $v'_{2\ell} = v_1$), $v_{2j} \in M_{2i},\ v_{2j+1} \notin M_{2i}$ for some $j \neq i$ implies $j = i+1$. Therefore, if $(w_i, x_i) \in \mathcal{S}_p$ then $(w_i, x_i) \in \{(u_{2i-4}, u_{2i-3}), (u_{2i-2}, u_{2i-1}), (u_{2i+1}, u_{2i}), (u_{2i+3}, u_{2i+2})\}$, and the same holds for (y_i, z_i). Note also that the pairs (w_i, x_i) and (z_i, y_i) play a symmetric role. Thus we can reduce by symmetries (on the sequence and on the two of (w_i, x_i) and (z_i, y_i)) to the six following cases:

- Case $x_i, y_i \notin \mathcal{S}_P$. See Fig 16. In particular, $w_i, z_i \notin \mathcal{S}_P$.

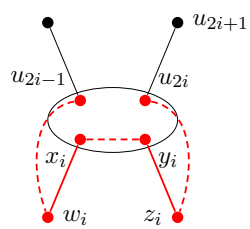

Figure 16: Case $x_i, y_i \notin \mathcal{S}_P$.

  We insert the F'-alternating subpath $(u_{2i-1}, w_i, x_i, y_i, z_i, u_{2i})$.

- Case $x_i \notin \mathcal{S}_P$, $y_i \in \{u_{2i-1}, u_{2i}\}$. We assume by symmetry $y_i = u_{2i}$. See Fig 17. In particular, we have $w_i \notin \mathcal{S}_P$.

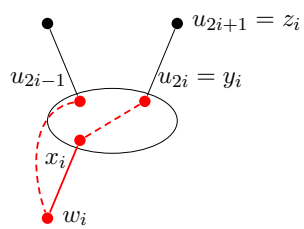

Figure 17: Case $x_i \notin \mathcal{S}_P$, $y_i = u_{2i}$.

  We insert the F'-alternating subpath $(u_{2i-1}, w_i, x_i, y_i = u_{2i})$. Note that the case $x_i \in \{u_{2i-1}, u_{2i}\}$, $y_i \notin \mathcal{S}_P$ is symmetrical to this one.

- Case $x_i \notin \mathcal{S}_P$, $y_i \in \mathcal{S}_p \setminus \{u_{2i-1}, u_{2i}\}$. We assume by symmetry $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$ (the case $(z_i, y_i) = (u_{2i-4}, u_{2i-3})$ is obtained by reverting the indices along the sequence). See Fig 18.

  We replace $(u_{2i-1}, u_{2i}, u_{2i+1}, y_i = u_{2i+2})$ by the F'-alternating subpath $(u_{2i-1}, w_i, x_i, y_i)$. Note that the case $x_i \in \mathcal{S}_p \setminus \{u_{2i-1}, u_{2i}\}$, $y_i \notin \mathcal{S}_P$ is symmetrical to this one.

- Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $y_i = u_{2i}$. See Fig 19.

  We replace $(u_{2i-3} = x_i, u_{2i-2}, u_{2i-1}, u_{2i} = y_i)$ by the F'-alternating subpath (x_i, y_i). Note that the case $x_i = u_{2i-1}$, $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$, and the two more cases obtained by switching the respective roles of (x_i, w_i) and (y_i, z_i), are symmetrical to this one.

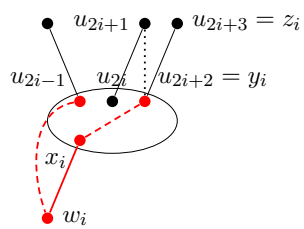

Figure 18: Case $x_i \notin \mathcal{S}_P$, $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$.

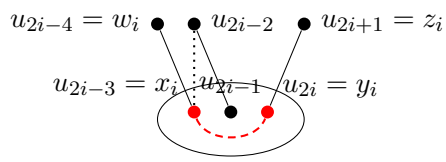

Figure 19: Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $y_i = u_{2i}$.

- Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $y_i = u_{2i-1}$. See Fig 20.

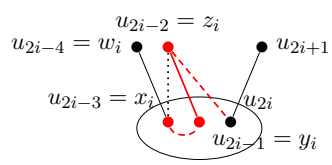

Figure 20: Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $y_i = u_{2i-1}$.

  We replace $(u_{2i-3} = x_i, u_{2i-2} = z_i, u_{2i-1} = y_i, u_{2i})$ by the F'-alternating subpath (x_i, y_i, z_i, u_{2i}). Note that the case $x_i = u_{2i}$, $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$, and the two more cases obtained by switching the respective roles of (x_i, w_i) and (y_i, z_i), are symmetrical to this one.

- Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$. See Fig 21.

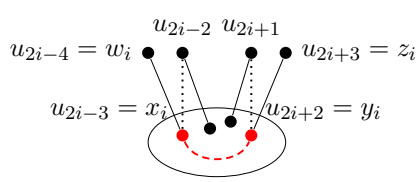

Figure 21: Case $(w_i, x_i) = (u_{2i-4}, u_{2i-3})$, $(y_i, z_i) = (u_{2i+2}, u_{2i+3})$.

  Since $x_i = u_{2i-3}$, $y_i = u_{2i+2}$ are adjacent we can remove $(u_{2i-2}, u_{2i-1}, u_{2i}, u_{2i+1})$ from $\mathcal{S}_P$.

Overall, in every case the procedure only depends on the subsequence between u_{2i-4} and u_{2i+3}. In order to prove correctness of the procedure, it suffices to prove that this subsequence has not been modified for a smaller $i' \in I_P$, $i' < i$. Equivalently, we prove that the above procedure does not modify the subsequence between u_{2j-4} and u_{2j+3} for any $j \in I_P$, $j > i$. First we claim $j \geq i + 2$. Indeed, $M_{2i+1} \in N_{G'}(M_{2i})$. Hence, by Lemma 14, $i \in I_P$ implies $i + 1 \notin I_P$, that proves the claim. In this situation, $2j - 4 \geq 2i$. Furthermore, the subsequence $(u_{2i}, \ldots, u_{2\ell})$ is modified only

if $u_{2i+2} \in \{x_i, y_i\}$. However in the latter case we have $u_{2i+3} \notin M_{2i}$, hence $M_{2i+3} \in N_{G'}(M_{2i})$, and so, by Lemma 14 $j \geq i+3$. In particular, $2j-4 \geq 2i+2$ and the subsequence $(u_{2i+2}, \ldots, u_{2\ell})$ is not modified by the procedure. Altogether, it proves that the above procedure is correct.

Finally, applying the above procedure for all $i \in I_P$ leads to an F'-alternating path in G'_F. □

We can now state the main result in this subsection.

Theorem 23. *For every $G = (V, E)$,* Maximum Matching *can be solved in $\mathcal{O}(mw(G)^4 \cdot n + m)$-time.*

Proof. The algorithm is recursive. If G is trivial (reduced to a single node) then we output an empty matching. Otherwise, let $G' = (\mathcal{M}(G), E')$ be the quotient graph of G. For every module $M \in \mathcal{M}(G)$, we call the algorithm recursively on $G[M]$ in order to compute a maximum matching F_M of $G[M]$. Let $F^* = \bigcup_{M \in \mathcal{M}(G)} F_M$. By Lemma 13 (applied to every $M \in \mathcal{M}(G)$ sequentially), we are left to compute a maximum matching for $G^* = (V, (E \setminus \bigcup_{M \in \mathcal{M}(G)} E(G[M])) \cup F^*)$. Therefore from now on assume $G = G^*$.

If G' is edgeless then we can output F^*. Otherwise, by Theorem 1 G' is either prime for modular decomposition or a complete graph.

Suppose G' to be prime. We start from $F_0 = F^*$. Furthermore, we ensure that the two following hold at every step $t \geq 0$:

- All the vertices that are matched in F^* are also matched in the current matching F_t. For instance, it is the case if F_t is obtained from F_0 by only using augmenting paths in order to increase the cardinality of the matching.
- For every $M \in \mathcal{M}(G)$ we store $|F_M \cap F_t|$. For every $M, M' \in \mathcal{M}(G)$ adjacent in G' we store $|(M \times M') \cap F_t|$. In particular, $|F_M \cap F_0| = |F_M|$ and $|(M \times M') \cap F_0| = 0$. So, it takes time $\mathcal{O}(\sum_{M \in \mathcal{M}(G)} deg_{G'}(M))$ to initialize this information, that is in $\mathcal{O}(|E(G')|) = \mathcal{O}(\mathsf{mw}(G)^2)$. Furthermore, it takes $\mathcal{O}(\ell)$-time to update this information if we increase the size of the matching with an augmenting path of length 2ℓ.

We construct the graph G'_{F_t} according to Definition 1. By using the information we store for the algorithm, it can be done in $\mathcal{O}(|E(G'_{F_t})|)$-time, that is in $\mathcal{O}(|E(G')|^2) = \mathcal{O}(\mathsf{mw}(G)^4)$ by Lemma 15. Furthermore by Theorem 22 there exists an F_t-augmenting path if and only if F_t is not maximum. Since we can assume all the modules in $\mathcal{M}(G)$ induce a matching, by Lemma 16 there exists an F_t-augmenting path in G if and only if there exists an F'_t-augmenting path in G'_{F_t}. So, we are left to compute an F'_t-augmenting path in G'_{F_t} if any. It can be done in $\mathcal{O}(|E(G'_{F_t})|)$-time [59], that is in $\mathcal{O}(\mathsf{mw}(G)^4)$. Furthermore, by construction of G'_{F_t}, an F'_t-augmenting path P' in G'_{F_t} is also an F_t-augmenting path in G. Thus, we can obtain a larger matching F_{t+1} from F_t and P. We repeat the procedure above for F_{t+1} until we reach a maximum matching $F_{t_{\max}}$. The total running time is in $\mathcal{O}(\mathsf{mw}(G)^4 \cdot t_{\max})$.

Finally, assume G' to be complete. Let $\mathcal{M}(G) = \{M_1, M_2, \ldots, M_k\}$ be linearly ordered. For every $1 \leq i \leq k$, write $G_i = G[\bigcup_{j \leq i} M_j]$. We compute a maximum matching F^i for G_i, from a maximum matching F^{i-1} of G_{i-1} and a maximum matching F_{M_i} of $G[M_i]$, sequentially. For that, we apply the same techniques as for the prime case, to some "pseudo-quotient graph" G'_i isomorphic to K_2 (*i.e.*, the two vertices of G'_i respectively represent $V(G_{i-1})$ and M_i). Since the pseudo-quotient graphs have size two, this step takes total time $\mathcal{O}(|V(G')| + (|F^k| - |F^*|))$.

Overall, summing the order of all the subgraphs in the modular decomposition of G amounts to $\mathcal{O}(n)$ [92]. Furthermore, a maximum matching of G also has cardinality $\mathcal{O}(n)$. Therefore, the total running time is in $\mathcal{O}(\mathsf{mw}(G)^4 \cdot n)$ if the modular decomposition of G is given. The latter decomposition can be precomputed in $\mathcal{O}(n+m)$-time [98]. □

5.2 More structure: $(q, q-3)$-graphs

The second main result in Section 5 is an $\mathcal{O}(\mathsf{q}(G)^4 \cdot n + m)$-time algorithm for Maximum Matching (Theorem 25). Our algorithm for $(q, q-3)$-graphs reuses the algorithm described in Theorem 23 as a subroutine. However, applying the same techniques to a case where the quotient graph has super-constant size $\Omega(q)$ happens to be more challenging. Thus we need to introduce new techniques in order to handle with all the cases presented in Lemma 4.

Computing a maximum matching for the *quotient graph* is easy. However, we also need to account for the edges present inside the modules. For that, we need the following stronger variant of Lemma 4. The latter generalizes similar structure theorems that have been obtained for some specific subclasses [65].

Theorem 24. *For an arbitrary $(q, q-3)$-graph G, $q \geq 7$, and its quotient graph G', exactly one of the following conditions is satisfied.*

1. *G is disconnected;*
2. *$\overline{G}$ is disconnected;*
3. *G is a disc (and so, $G = G'$ is prime for modular decomposition);*
4. *G is a spider (and so, G' is a prime spider);*
5. *G' is a spiked p-chain P_k, or a spiked p-chain $\overline{P_k}$. Furthermore, for every $v \in V(G')$, if the corresponding module $M_v \in \mathcal{M}(G)$ is such that $|M_v| \geq 2$ then we have $v \in \{v_1, v_k, x, y\}$;*
6. *G' is a spiked p-chain Q_k, or a spiked p-chain $\overline{Q_k}$. Furthermore, for every $v \in V(G')$, if the corresponding module $M_v \in \mathcal{M}(G)$ is such that $|M_v| \geq 2$ then we have either $v \in \{v_1, v_k\}$ or $v = z_i$ for some i;*
7. *$|V(G')| \leq q$.*

The proof of Theorem 24 is postponed to the appendix. It is based on a refinement of modular decomposition called *primeval decomposition*.

In what follows, we introduce our techniques for the cases where the quotient graph G' is neither degenerate nor of constant size.

Simple cases

Lemma 17. *For every disc G, a maximum matching can be computed in linear time.*

Proof. If $G = C_n$, $n \geq 5$ is a cycle then the set of edges $\{\{2i, 2i+1\} \mid 0 \leq i \leq \lfloor n/2 \rfloor - 1\}$ is a maximum matching. Otherwise, $G = \overline{C_n}$ is a co-cycle. Let F_n contain all the edges $\{4i, 4i+2\}$, $\{4i+1, 4i+3\}$, $0 \leq i \leq \lfloor n/4 \rfloor - 1$. There are three cases. If $n \equiv 0 \pmod 4$ or $n \equiv 1 \pmod 4$ then there is at most one vertex unmatched by F_n, and so, F_n is a maximum matching of G. Otherwise, if

$n \equiv 3 \pmod 4$ then a maximum matching of G is obtained by adding the edge $\{n-3, n-1\}$ to F_n. Finally, assume $n \equiv 2 \pmod 4$. By construction, F_n leaves unmatched the two of $n-2, n-1$. We obtain a perfect matching of G from F_n by replacing $\{0, 2\}$ with $\{n-2, 0\}$, $\{n-1, 2\}$. Note that it is possible to do that since $n \geq 5$. □

Lemma 18. *If $G = (S \cup K \cup R, E)$ is a spider then there exists a maximum matching of G composed of: a perfect matching between K and S; and a maximum matching of $G[R]$.*

Proof. We start from a perfect matching F_0 between K and S. We increase the size of F_0 using augmenting paths until it is no more possible to do so. By Theorem 22, the obtained matching $F_{\max}$ is maximum. Furthermore, either there is a perfect matching between K and S or there is at least one vertex of S that is unmatched. Since $V(F_0) \subseteq V(F_{\max})$ the latter proves the lemma. □

The case of prime p-trees

Roughly, when the quotient graph G' is a prime p-tree, our strategy consists in applying the following reduction rules until the graph is empty.

1. Find an isolated module M (with no neighbour). Compute a maximum matching for $G[M]$ and for $G[V \setminus M]$ separately.

2. Find a pending module M (with one neighbour v). Compute a maximum matching for $G[M]$. If it is not a perfect matching then add an edge between v and any unmatched vertex in M, then discard $M \cup \{v\}$. Otherwise, discard M (Lemma 19).

3. Apply a technique known as "SPLIT and MATCH" [101] to some module M and its neighbourhood $N_G(M)$. We do so only if M satisfies some properties. In particular, we apply this rule when M is a universal module (with a complete join between M and $V \setminus M$). See Definition 2 and Lemma 20.

We introduce the reduction rules below and we prove their correctness.

Reduction rules. The following lemma generalizes a well-known reduction rule for Maximum Matching: add a pending vertex and its unique neighbour to the matching then remove this edge [78].

Lemma 19. *Let M be a module in a graph $G = (V, E)$ such that $N_G(M) = \{v\}$, F_M is a maximum matching of $G[M]$ and F^*_M is obtained from F_M by adding an edge between v and any unmatched vertex of M (possibly, $F^*_M = F_M$ if it is a perfect matching). There exists a maximum matching F of G such that $F^*_M \subseteq F$.*

Proof. By Lemma 13, every maximum matching for $G'_M = (V, (E \setminus E(G[M]) \cup F_M)$ is also a maximum matching for G. There are two cases.

Suppose there exists $u \in M \setminus V(F_M)$. Then, u is a pending vertex of G'_M. There exists a maximum matching of G'_M that contains the edge $\{u, v\}$ [78]. Furthermore, removing u and v disconnects the vertices of $M \setminus u$ from $V \setminus N_G[M]$. It implies that a maximum matching F' of $G \setminus (u, v)$ is the union of any maximum matching of $G[M \setminus u]$ with any maximum matching of $G[V \setminus N_G[M]]$. In particular, F_M is contained in some maximum matching F' of $G \setminus (u, v)$. Since

$\{u, v\}$ is contained in a maximum matching of G, therefore $F = F' \cup \{\{u, v\}\}$ is a maximum matching of G. We are done since $F^*_M = F_M \cup \{\{u, v\}\} \subseteq F$ by construction.

Otherwise, F_M is a perfect matching of $G[M]$. For every edge $\{x, y\} \in F_M$, we have that x, y have degree two in G'_M. The following reduction rule has been proved to be correct in [78]: remove any x of degree two, merge its two neighbours and increase the size of the solution by one unit. In our case, since $N_{G'_M}[y] \subseteq N_{G'_M}[v]$ the latter is equivalent to put the edge $\{x, y\}$ in the matching. Overall, applying the reduction rule to all edges $\{x, y\} \in F_M$ indeed proves the existence of some maximum matching F such that $F_M = F^*_M \subseteq F$. □

Then, we introduce a technique known as "SPLIT and MATCH" in the literature [101].

Definition 2. Let $G = (V, E)$ be a graph, $F \subseteq E$ be a matching of G. Given some module $M \in \mathcal{M}(G)$ we try to apply the following two operations until none of them is possible:

- Suppose there exist $u \in M,\ v \in N_G(M)$ unmatched. We add an edge $\{u, v\}$ to the matching (MATCH).
- Otherwise, suppose there exist $u, u' \in M,\ v, v' \in N_G(M)$ such that u and u' are unmatched, and $\{v, v'\}$ is an edge of the matching. We replace the edge $\{v, v'\}$ in the matching by the two new edges $\{u, v\},\ \{u', v'\}$ (SPLIT).

The "SPLIT and MATCH" has been applied to compute a maximum matching in linear time for cographs and some of its generalizations [54, 55, 101]. Our Theorem 23 can be seen as a broad generalization of this technique. In what follows, we introduce more cases where the "SPLIT and MATCH" technique can be used in order to compute a maximum matching directly.

Lemma 20. *Let $G = G_1 \oplus G_2$ be the join of two graphs G_1, G_2 and let F_1, F_2 be maximum matchings for G_1, G_2, respectively. For $F = F_1 \cup F_2$, applying the "*SPLIT *and* MATCH*" technique to $V(G_1)$, then to $V(G_2)$ leads to a maximum matching of G.*

Proof. The lemma is proved in [101] when G is a cograph. In particular, let $G^* = (V, (V(G_1) \times V(G_2)) \cup F_1 \cup F_2)$. Since it ignores the edges from $(E(G_1) \setminus F_1) \cup (E(G_2) \setminus F_2)$, the procedure outputs the same matching for G and G^*. Furthermore, G^* is a cograph, and so, the outputted matching is maximum for G^*. By Lemma 13, a maximum matching for G^* is a maximum matching for G. □

Applications. We can now combine our reductions rules as follows.

Proposition 1. *Let $G = (V, E)$ be a $(q, q-3)$-graph, $q \geq 7$, such that its quotient graph G' is isomorphic to a prime p-tree. For every $M \in \mathcal{M}(G)$ let F_M be a maximum matching of $G[M]$ and let $F^* = \bigcup_{M \in \mathcal{M}(G)} F_M$.*

A maximum matching $F_{\max}$ for G can be computed in $\mathcal{O}(|V(G')| + |E(G')| + [|F_{\max}| - |F^*|])$-time if F^* is given as part of the input.

Proof. There are five cases. If G' has order at most 7 then we can apply the same techniques as for Theorem 23. Otherwise, G' is either a spiked p-chain P_k, a spiked p-chain $\overline{P_k}$, a spiked p-chain Q_k or a spiked p-chain $\overline{Q_k}$.

Case G is a spiked p-chain P_k. By Theorem 24 we have that $(v_2, v_3, \ldots, v_{k-1})$ are vertices of G. In this situation, since $N_{G'}(v_1) = v_2$, M_{v_1} is a pending module. We can apply the reduction rule of Lemma 19 to M_{v_1}. Doing so, we discard M_{v_1} and possibly v_2. Let $S = M_x$ if v_2 has already been discarded and let $S = M_x \cup \{v_2\}$ otherwise. We have that S is a pending module in the resulting subgraph, with v_3 being its unique neighbour. Furthermore, by Lemma 19 we can compute a maximum matching of $G[S]$ from $F_{M_{v_x}}$, by adding an edge between v_2 (if it is present) and an unmatched vertex in M_x (if any). So, we again apply the reduction rule of Lemma 19, this time to S. Doing so, we discard S, and possibly v_3. Then, by a symmetrical argument we can also discard M_{v_k}, M_y, v_{k-1} and possibly v_{k-2}. We are left with computing a maximum matching for some subpath of $(v_3, v_4, \ldots, v_{k-2})$, that can be done in linear time by taking half of the edges.

Case G is a spiked p-chain $\overline{P_k}$. By Theorem 24, the nontrivial modules of $\mathcal{M}(G)$ can only be $M_{v_1}, M_{v_k}, M_x, M_y$. In particular, $F^* = F_{M_{v_1}} \cup F_{M_{v_k}} \cup F_{M_x} \cup F_{M_y}$. Let $U = M_{v_1} \cup M_{v_k} \cup M_x \cup M_y$. The graph $G \setminus U$ is isomorphic to $\overline{P_{k-2}}$, $k \geq 6$. Furthermore, let F_{k-2} contain the edges $\{v_2, v_{\lceil k/2 \rceil + 1}\}$, $\{v_{\lfloor k/2 \rfloor}, v_{k-1}\}$ plus all the edges $\{v_i, v_{k+1-i}\}$, $3 \leq i \leq \lfloor k/2 \rfloor - 1$. Observe that F_{k-2} is a maximum matching of $\overline{P_{k-2}}$. In particular it is a perfect matching of $\overline{P_{k-2}}$ if k is even, and if k is odd then it only leaves vertex $v_{\lceil k/2 \rceil}$ unmatched. We set $F_0 = F^* \cup F_{k-2}$ to be the initial matching. Then, we repeat the procedure below until we cannot increase the matching anymore. We consider the modules $M \in \{M_{v_1}, M_{v_k}, M_x, M_y\}$ sequentially. For every M we try to apply the SPLIT and MATCH technique of Definition 2.

Overall, we claim that the above procedure can be implemented to run in constant time per loop. Indeed, assume that the matched vertices (resp., the unmatched vertices) are stored in a list in such a way that all the vertices in a same module M_v, $v \in V(G')$ are consecutive. For every matched vertex u, we can access to the vertex that is matched with u in constant time. Furthermore for every $v \in V(G')$, we keep a pointer to the first and last vertices of M_v in the list of matched vertices (resp., in the list of unmatched vertices). For any loop of the procedure, we iterate over four modules M, that is a constant. Furthermore, since $|N_G(M)| \geq |V(G) \setminus M| - 2$ then we only need to check three unmatched vertices of $V \setminus M$ in order to decide whether we can perform a MATCH operation. Note that we can skip scanning the unmatched vertices in M using our pointer structure, so, it takes constant time. In the same way, we only need to consider three matched vertices of $V \setminus M$ in order to decide whether we can perform a SPLIT operation. Again, it takes constant time. Therefore, the claim is proved.

Let $F_{\max}$ be the matching so obtained. By the above claim it takes $\mathcal{O}(|F_{\max}| - |F_0|)$-time to compute it with the above procedure. Furthermore, we claim that $F_{\max}$ is maximum. Suppose for the sake of contradiction that $F_{\max}$ is not a maximum matching. By Lemma 13, $F_{\max}$ cannot be a maximum matching of G^*, obtained from G by removing the edges in $(E(G[M_{v_1}]) \cup E(G[M_{v_x}]) \cup E(G[M_{v_y}]) \cup E(G[M_{v_k}])) \setminus F^*$. Let $P = (u_1, u_2, \ldots, u_{2\ell})$ be a *shortest* $F_{\max}$-augmenting path in G^*, that exists by Theorem 22.

We prove as an intermediate subclaim that both $u_1, u_{2\ell}$ must be part of a same module amongst $M_{v_1}, M_{v_k}, M_x, M_y$. Indeed, for every distinct $M, M' \in \{M_{v_1}, M_{v_k}, M_x, M_y\}$, every vertex of M is adjacent to every vertex of M'. Furthermore, $V(F_{k-2}) \subseteq V(F_{\max})$ by construction and $v_{\lceil k/2 \rceil}$ (the only vertex of $\overline{P_{k-2}}$ possibly unmatched) is adjacent to every vertex of U. Therefore, if the subclaim were false then $u_1, u_{2\ell}$ should be adjacent, hence they should have been matched together with a MATCH operation. A contradiction. So, the subclaim is proved.

Let $M \in \{M_{v_1}, M_{v_k}, M_x, M_y\}$ so that $u_1, u_{2\ell} \in M$. Since $E(G^*[M]) = F_M$, and $V(F_M) \subseteq$

$V(F^*) \subseteq V(F_{\max})$ by construction, we have $u_2 \in N_G(M)$. Furthermore, $u_3 \notin N_G(M)$ since otherwise, by considering $u_1, u_{2\ell} \in M$ and $u_2, u_3 \in N_G(M)$, we should have increased the matching with a SPLIT operation. In this situation, either $u_3 \in M$ or $u_3 \in V \setminus N_G[M]$. We prove as another subclaim that $u_3, u_4 \in M$. Indeed, suppose by contradiction $u_4 \in N_G(M)$. In particular, $(u_1, u_4, u_5, \ldots, u_{2\ell})$ is a shorter augmenting path than P, thereby contradicting the minimality of P. Therefore, $u_4 \notin N_G(M)$. Moreover, if $u_3 \in V \setminus N_G[M]$ then, since the set $V \setminus N_G[M]$ induces a stable, we should have $u_4 \in N_G(M)$. A contradiction. So, $u_3 \in M$, and $u_4 \in N_G[M] \setminus N_G(M) = M$, that proves the subclaim.

The above subclaim implies $\{u_3, u_4\} \in F_M$. Since $\{u_3, u_4\} \notin F_{\max}$, there exists a module M' such that, during some loop of the algorithm, the edge $\{u_3, u_4\}$ has been replaced with two edges $\{w_2, u_3\}$ and $\{u_4, w_5\}$, for some $w_2, w_5 \in M'$ (possibly, $w_2 = u_2$ or $w_5 = u_5$) by performing a SPLIT operation. However, since $u_1, u_{2\ell} \in M$ are unmatched, and $M \subseteq N_G(M')$, we should have performed two MATCH operations intead of performing a SPLIT operation. A contradiction. Therefore, as claimed, $F_{\max}$ is a maximum matching of G.

Case G is a spiked p-chain Q_k. For every $1 \leq i \leq \lceil k/2 \rceil$, let $V_i = \bigcup_{j \geq i} (M_{v_{2j-1}} \cup M_{v_{2j}} \cup M_{z_{2j-1}} \cup M_{z_{2j}})$ (by convention $M_v = \emptyset$ if vertex v is not present). Roughly, our algorithm tries to compute recursively a maximum matching for $G_i = G[V_i \cup U_{i-1}]$, where U_{i-1} is a union of modules in $\{M_{v_{2i-2}},\ M_{z_{2i-2}}\}$. Initially, we set $i = 1$ and $U_0 = \emptyset$. See Fig. 22.

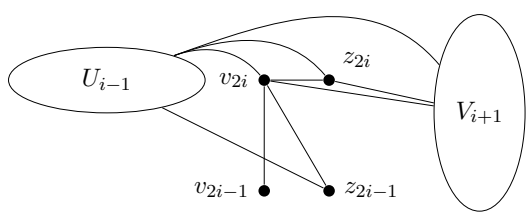

Figure 22: Schematic view of graph G_i.

If $i = \lceil k/2 \rceil$ then the quotient subgraph G'_i has order at most six. We can reuse the same techniques as for Theorem 23 in order to solve this case. Thus from now on assume $i < \lceil k/2 \rceil$. We need to observe that v_{2i-1} is a pending vertex in the quotient subgraph G'_i, with v_{2i} being its unique neighbour. By Theorem 24, $v_{2i} \in V(G)$, hence $M_{v_{2i-1}}$ is a pending module of G_i. Thus, we can apply the reduction rule of Lemma 19. Doing so, we can discard the set S_i, where $S_i = M_{v_{2i-1}} \cup \{v_{2i}\}$ if $F_{M_{v_{2i-1}}}$ is not a perfect matching of $G[M_{v_{2i-1}}]$, and $S_i = M_{v_{2i-1}}$ otherwise.

Furthermore, in the case where $U_{i-1} \neq \emptyset$, there is now a complete join between U_{i-1} and $V_i \setminus S_i$. By Lemma 20 we can compute a maximum matching of $G_i \setminus S_i$ from a maximum matching of $G[U_{i-1}]$ and a maximum matching of $G[V_i \setminus S_i]$. In particular, since U_{i-1} is a union of modules in $\{M_{v_{2i-2}},\ M_{z_{2i-2}}\}$ and there is a complete join between $M_{v_{2i-2}}$ and $M_{z_{2i-2}}$, by Lemma 20 a maximum matching of $G[U_{i-1}]$ can be computed from $F_{M_{v_{2i-2}}}$ and $F_{M_{z_{2i-2}}}$. So, we are left to compute a maximum matching of $G[V_i \setminus S_i]$.

Then, there are two subcases. If $v_{2i} \in S_i$ then $M_{z_{2i-1}}$ is disconnected in $G[V_i \setminus S_i]$. Let $U_i = M_{z_{2i}}$. The union of $F_{M_{z_{2i-1}}}$ with a maximum matching of $G_{i+1} = G[V_{i+1} \cup U_i]$ is a maximum matching of $G[V_i \setminus S_i]$. Otherwise, $M_{z_{2i-1}}$ is a pending module of $G[V_i \setminus S_i]$ with v_{2i} being its unique neighbour. We apply the reduction rule of Lemma 19. Doing so, we can discard the set T_i, where

$T_i = M_{z_{2i-1}} \cup \{v_{2i}\}$ if $F_{M_{z_{2i-1}}}$ is not a perfect matching of $G[M_{z_{2i-1}}]$, and $T_i = M_{z_{2i-1}}$ otherwise. Let $U_i = M_{z_{2i}}$ if $v_{2i} \in T_i$ and $U_i = M_{z_{2i}} \cup M_{v_{2i}}$ otherwise. We are left to compute a maximum matching of $G_{i+1} = G[V_{i+1} \cup U_i]$. Overall, the procedure stops after we reach an empty subgraph, that takes $\mathcal{O}(|V(G')|)$ recursive calls.

Case G is a spiked p-chain $\overline{Q_k}$. Roughly, the case where G' is isomorphic to a spiked p-chain $\overline{Q_k}$ is obtained by reverting the role of vertices with even index and vertices with odd index. For every $1 \leq i \leq \lfloor k/2 \rfloor$, let $V_i = \bigcup_{j \geq i}(M_{v_{2j}} \cup M_{z_{2j}} \cup M_{v_{2j+1}} \cup M_{z_{2j+1}})$. Our algorithm tries to compute recursively a maximum matching for $G_i = G[V_i \cup U_{i-1}]$, where U_{i-1} is a union of modules in $\{M_{v_{2i-1}},\ M_{z_{2i-1}}\}$. Initially, we set $i = 1$ and $U_0 = M_{v_1}$.

If $i = \lfloor k/2 \rfloor$ then the quotient subgraph G'_i has order at most six. We can reuse the same techniques as for Theorem 23 in order to solve this case. Thus from now on assume $i < \lfloor k/2 \rfloor$. We need to observe that v_{2i} is a pending vertex in the quotient subgraph G'_i, with v_{2i+1} being its unique neighbour. By Theorem 24, $v_{2i+1} \in V(G)$, hence $M_{v_{2i}}$ is a pending module of G_i. Thus, we can apply the reduction rule of Lemma 19. Doing so, we can discard the set S_i, where $S_i = M_{v_{2i}} \cup \{v_{2i+1}\}$ if $F_{M_{v_{2i}}}$ is not a perfect matching of $G[M_{v_{2i}}]$, and $S_i = M_{v_{2i}}$ otherwise.

Furthermore, in the case where $U_{i-1} \neq \emptyset$, there is now a complete join between U_{i-1} and $V_i \setminus S_i$. By Lemma 20 we can compute a maximum matching of $G_i \setminus S_i$ from a maximum matching of $G[U_{i-1}]$ and a maximum matching of $G[V_i \setminus S_i]$. In particular, since U_{i-1} is a union of modules in $\{M_{v_{2i-1}},\ M_{z_{2i-1}}\}$ and there is a complete join between $M_{v_{2i-1}}$ and $M_{z_{2i-1}}$, by Lemma 20 a maximum matching of $G[U_{i-1}]$ can be computed from $F_{M_{v_{2i-2}}}$ and $F_{M_{z_{2i-2}}}$. So, we are left to compute a maximum matching of $G[V_i \setminus S_i]$.

Then, there are two subcases. If $v_{2i+1} \in S_i$ then $M_{z_{2i}}$ is disconnected in $G[V_i \setminus S_i]$. Let $U_i = M_{z_{2i+1}}$. The union of $F_{M_{z_{2i}}}$ with a maximum matching of $G_{i+1} = G[V_{i+1} \cup U_i]$ is a maximum matching of $G[V_i \setminus S_i]$. Otherwise, $M_{z_{2i}}$ is a pending module of $G[V_i \setminus S_i]$ with v_{2i+1} being its unique neighbour. We apply the reduction rule of Lemma 19. Doing so, we can discard the set T_i, where $T_i = M_{z_{2i}} \cup \{v_{2i+1}\}$ if $F_{M_{z_{2i}}}$ is not a perfect matching of $G[M_{z_{2i}}]$, and $T_i = M_{z_{2i}}$ otherwise. Let $U_i = M_{z_{2i+1}}$ if $v_{2i+1} \in T_i$ and $U_i = M_{z_{2i+1}} \cup M_{v_{2i+1}}$ otherwise. We are left to compute a maximum matching of $G_{i+1} = G[V_{i+1} \cup U_i]$. Overall, the procedure stops after $\mathcal{O}(|V(G')|)$ recursive calls. □

Main result

Theorem 25. *For every graph G,* Maximum Matching *can be solved in $\mathcal{O}(\mathsf{q}(G)^4 \cdot n + m)$-time.*

Proof. We generalize the algorithm for Theorem 23. In particular the algorithm is recursive. If G is trivial (reduced to a single node) then we output an empty matching. Otherwise, let $G' = (\mathcal{M}(G), E')$ be the quotient graph of G. For every module $M \in \mathcal{M}(G)$, we call the algorithm recursively on $G[M]$ in order to compute a maximum matching F_M of $G[M]$. Let $F^* = \bigcup_{M \in \mathcal{M}(G)} F_M$. If G' is either edgeless, complete or a prime graph with no more than $\mathsf{q}(G)$ vertices then we apply the same techniques as for Theorem 23 in order to compute a maximum matching $F_{\max}$ for G. It takes constant time if G' is a stable, $\mathcal{O}(\mathsf{q}(G)^4 \cdot (|F_{\max}| - |F^*|))$-time if G' is prime and $\mathcal{O}(|V(G')| + (|F_{\max}| - |F^*|))$-time if G' is a complete graph. Otherwise by Theorem 24 the following cases need to be considered.

- Suppose G is a disc. In particular, $G = G'$. By Lemma 17, we can compute a maximum matching for G in $\mathcal{O}(|V(G')| + |E(G')|)$-time.

- Suppose $G = (S \cup K \cup R, E)$ is a spider. In particular, $G' = (S \cup K \cup R', E')$ is a prime spider. By Lemma 18, the union of $F_R = F^*$ with a perfect matching between S and K is a maximum matching of G. It can be computed in $\mathcal{O}(|V(G')| + |E(G')|)$-time.

- Otherwise G' is a prime p-tree. By Proposition 1, a maximum matching $F_{\max}$ for G can be computed in $\mathcal{O}(|V(G')| + |E(G')| + [|F_{\max}| - |F^*|])$-time.

Overall, summing the order of all the subgraphs in the modular decomposition of G amounts to $\mathcal{O}(n)$ [92]. Summing the size of all the subgraphs in the modular decomposition of G amounts to $\mathcal{O}(n+m)$ [92]. Furthermore, a maximum matching of G also has cardinality $\mathcal{O}(n)$. Therefore, the total running time is in $\mathcal{O}(\mathsf{q}(G)^4 \cdot n + m)$ if the modular decomposition of G is given. The latter decomposition can be precomputed in $\mathcal{O}(n+m)$-time [98]. □

6 Applications to other graph classes

Our algorithmic schemes in Sections 4 and 5 are all based on preprocessing methods with either split decomposition or modular decomposition. If the prime subgraphs of the decomposition have constant-size then the input graph has bounded clique-width. However, when the prime subgraphs are "simple" enough w.r.t. the problem considered, we may well be able to generalize our techniques in order to apply to some graph classes with unbounded clique-width. In what follows, we present such examples.

A graph is *weak bipolarizable* if every prime subgraph in its modular decomposition is a chordal graph [87]. Some cycle problems such as Girth (trivially) and Triangle Counting (by using a clique-tree) can be easily solved in linear time for chordal graphs. The latter extends to the larger class of weak bipolarizable graphs by using our techniques.

Another instructive example is the class of graphs with small prime subgraphs for *c-decomposition*. The c-decomposition consists in successively decomposing a graph by the modular decomposition and the split decomposition until all the subgraphs obtained are either degenerate (complete, edgeless or star) or prime for both the modular decomposition and the split decomposition [82]. Let us call c-width the minimum $k \geq 2$ such that any prime subgraph in the c-decomposition has order at most k. The following was proved in [91].

Theorem 26 ([91])**.** *The class of graphs with c-width 2 (i.e., completely decomposable by the c-decomposition) has unbounded clique-width.*

It is not clear how to compute the c-decomposition in linear time. However, both the modular decomposition and the split decomposition of graphs with small c-width already have some interesting properties which can be exploited for algorithmic purposes. Before concluding this section we illustrate this fact with Eccentricities.

Lemma 21. *Let G be a graph with c-width at most k that is prime for modular decomposition. Every split component of G that is not degenerate either has order at most k or contains a universal vertex.*

Proof. Since G has c-width at most k, every non degenerate split component of G with order at least $k+1$ can be modularly decomposed. We show in the proof of Lemma 11 that if a non degenerate graph can be modularly decomposed and it does not contain a universal vertex then it has a split.

Therefore, every non degenerate split component of size at least $k+1$ contains a universal vertex since it is prime for split decomposition. $\square$

We now revisit the algorithmic scheme of Theorem 9.

Proposition 2. *For every graph G with c-width at most k,* Eccentricities *can be solved in $\mathcal{O}(k^2 \cdot n + m)$-time. In particular,* Diameter *can also be solved in $\mathcal{O}(k^2 \cdot n + m)$-time.*

Proof. Let $G' = (V', E')$ be the quotient graph of G. Note that G' has c-width at most k. Furthermore, by Theorem 14 the problem reduces in linear time to solve Eccentricities for G'. We compute the split-decomposition of G'. It takes linear time [22]. By Lemma 21 every split component of G' either has order at most k or it has diameter at most 2.

Let us consider the following subproblem for every split component C_i. Given a weight function $e : V(C_i) \to \mathbb{N}$, compute $\max_{u \in V(C_i) \setminus \{v\}} dist_{C_i}(u, v) + e(u)$ for every $v \in C_i$. Indeed, the algorithm for Theorem 9 consists in solving the above subproblem a constant-number of times for every split component, with different weight functions e that are computed by tree traversal on the split decomposition tree. In particular, if the above subproblem can be solved in $\mathcal{O}(k^2 \cdot |V(C_i)| + |E(C_i)|)$-time for every split component C_i then we can solve Eccentricities for G' in $\mathcal{O}(k^2 \cdot |V(G')| + |E(G')|)$-time.

There are two cases. If C_i has order at most k then the above subproblem can be solved in $\mathcal{O}(|V(C_i)||E(C_i)|)$-time, that is in $\mathcal{O}(k^2 \cdot |V(C_i)|)$. Otherwise, by Lemma 21 C_i contains a universal vertex, that can be detected in $\mathcal{O}(|V(C_i)| + |E(C_i)|)$-time. In particular, C_i has diameter at most two. Let $V(C_i) = (v_1, v_2, \ldots, v_{|V(C_i)|})$ be totally ordered such that, for every $j < j'$ we have $e(v_j) \geq e(v_{j'})$. An ordering as above can be computed in $\mathcal{O}(|V(C_i)|)$-time, for instance using a bucket-sort algorithm. Then, for every $v \in V(C_i)$ we proceed as follows. We compute $D_v = 1 + \max_{u \in N_{C_i}(v)} e(u)$. It takes $\mathcal{O}(deg_{C_i}(v))$-time. Then, we compute the smallest j such that v_j and v are nonadjacent (if any). Starting from v_1 and following the ordering, it takes $\mathcal{O}(deg_{C_i}(v))$-time. Finally, we are left to compare, in constant time, D_v with $2 + e(v_i)$. Overall, the subproblem is solved in $\mathcal{O}(|V(C_i)| + |E(C_i)|)$-time in this case.

Therefore, Eccentricities can be solved in $\mathcal{O}(k^2 \cdot n + m)$-time for G. $\square$

Acknowledgements

We wish to thank the referees for their careful reading of the first version of this manuscript, and their useful comments. This work has been supported by the French government, through the UCA^{JEDI} Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-01. This work was also supported by the Institutional research programme PN 1819 "Advanced IT resources to support digital transformation processes in the economy and society - RESINFO-TD" (2018), project PN 1819-01-01"Modeling, simulation, optimization of complex systems and decision support in new areas of IT&C research", funded by the Ministry of Research and Innovation, Romania, and by a grant of Romanian Ministry of Research and Innovation CCCDI-UEFISCDI. project no. 17PCCDI/2018.

References

[1] A. Abboud, F. Grandoni, and V. Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1681–1697, Philadelphia, PA, USA, 2015. SIAM.

[2] A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443. IEEE, 2014.

[3] A. Abboud, V. Vassilevska Williams, and J. R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016.

[4] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[5] L. Babel. Tree-like P_4-connected graphs. *Discrete Mathematics*, 191(1-3):13–23, 1998.

[6] L. Babel. Recognition and isomorphism of tree-like P_4-connected graphs. *Discrete Applied Mathematics*, 99(1):295–315, 2000.

[7] L. Babel and S. Olariu. On the structure of graphs with few P_4's. *Discrete Applied Mathematics*, 84(1–3):1–13, 1998.

[8] L. Babel and S. Olariu. On the p-connectedness of graphs – a survey. *Discrete Applied Mathematics*, 95(1-3):11–33, 1999.

[9] H.-J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[10] S. Baumann. A linear algorithm for the homogeneous decomposition of graphs. Technical Report M-9615, Zentrum Mathematik, Technische Universität München, 1996.

[11] M. Bentert, T. Fluschnik, A. Nichterlein, and R. Niedermeier. Parameterized aspects of triangle enumeration. In *International Symposium on Fundamentals of Computation Theory (FCT)*, volume 10472 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2017.

[12] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

[13] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1):1–45, 1998.

[14] H. L. Bodlaender. Treewidth: Characterizations, Applications, and Computations. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[15] H. L. Bodlaender and A. M. C. A. Koster. Safe separators for treewidth. *Discrete Mathematics*, 306(3):337–350, 2006.

[16] J. A. Bondy and U. S. R. Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag London, 2008.

[17] M. Borassi, D. Coudert, P. Crescenzi, and A. Marino. On computing the hyperbolicity of real-world graphs. In *European Symposia on Algorithms (ESA)*, volume 9294 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2015.

[18] M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.

[19] U. Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[20] G. Brinkmann, J. H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.

[21] B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. Boolean-width of graphs. *Theoretical computer science*, 412(39):5187–5204, 2011.

[22] P. Charbit, F. De Montgolfier, and M. Raffinot. Linear time split decomposition revisited. *SIAM Journal on Discrete Mathematics*, 26(2):499–514, 2012.

[23] N. Cohen, D. Coudert, G. Ducoffe, and A. Lancin. Applying clique-decomposition for computing gromov hyperbolicity. *Theoretical computer science*, 690:114–139, 2017.

[24] N. Cohen, D. Coudert, and A. Lancin. On computing the gromov hyperbolicity. *Journal of Experimental Algorithmics*, 20:1–6, 2015.

[25] D. G. Corneil, M. Habib, J.-M. Lanlignel, B. Reed, and U. Rotics. Polynomial Time Recognition of Clique-Width ≤ 3 Graphs. In *Latin American Theoretical INformatics Symposium (LATIN)*, volume 1776 of *Lecture Notes in Computer Science*, pages 126–134. Springer, 2000.

[26] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[27] D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.

[28] D. Coudert and G. Ducoffe. Recognition of C_4-free and 1/2-hyperbolic graphs. *SIAM Journal on Discrete Mathematics*, 28(3):1601–1617, 2014.

[29] D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2765–2784. SIAM, 2018.

[30] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[31] B. Courcelle. On the model-checking of monadic second-order formulas with edge set quantifications. *Discrete Applied Mathematics*, 160(6):866–887, 2012.

[32] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

[33] B. Courcelle, P. Heggernes, D. Meister, C. Papadopoulos, and U. Rotics. A characterisation of clique-width through nested partitions. *Discrete Applied Mathematics*, 187:70–81, 2015.

[34] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[35] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000.

[36] W. H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.

[37] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2010. *4th* edition.

[38] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer Science & Business Media, 1999.

[39] F. F. Dragan. On greedy matching ordering and greedy matchable graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 1335 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 1997.

[40] F. F. Dragan and F. Nicolai. LexBFS-orderings of distance-hereditary graphs with application to the diametral pair problem. *Discrete Applied Mathematics*, 98(3):191–207, 2000.

[41] R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM*, 61(1):1, 2014.

[42] G. Ducoffe. *Metric properties of large graphs*. PhD thesis, Université Côte d'Azur, Dec. 2016.

[43] G. Ducoffe and A. Popa. The b-matching problem in distance-hereditary graphs and beyond. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 123 of *Leibniz International Proceedings in Informatics*, pages 30:1–30:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[44] G. Ducoffe and A. Popa. The use of a pruned modular decomposition for maximum matching algorithms on some graph classes. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 123 of *Leibniz International Proceedings in Informatics*, pages 6:1–6:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[45] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[46] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 1 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2001.

[47] J. Evald and S. Dahlgaard. Tight hardness results for distance and centrality problems in constant degree graphs. *CoRR*, abs/1609.08403, 2016.

[48] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width is NP-complete. *SIAM Journal on Discrete Mathematics*, 23(2):909–939, 2009.

[49] T. Fluschnik, C. Komusiewicz, G. B. Mertzios, A. Nichterlein, R. Niedermeier, and N. Talmon. When can graph hyperbolicity be computed in linear time? In *Workshop on Algorithms and Data Structures*, volume 10389 of *Lecture Notes in Computer Science*, pages 397–408. Springer, 2017.

[50] F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010.

[51] F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.

[52] F. V. Fomin, P. A. Golovach, D. Lokshtanov, S. Saurabh, and M. Zehavi. Clique-width III: Hamiltonian Cycle and the Odd Case of Graph Coloring. *ACM Transactions on Algorithms*, 15(1):9, 2019.

[53] F. V. Fomin, D. Lokshtanov, S. Saurabh, M. Pilipczuk, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms*, 14(3):34:1–34:45, 2018.

[54] J.-L. Fouquet, V. Giakoumakis, and J.-M. Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International Journal of Foundations of Computer Science*, 10(4):513–533, 1999.

[55] J.-L. Fouquet, I. Parfenoff, and H. Thuillier. An $O(n)$-time algorithm for maximum matching in P_4-tidy graphs. *Information Processing Letter*, 62(6):281–287, 1997.

[56] H. Fournier, A. Ismail, and A. Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letter*, 115(6):576–579, 2015.

[57] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[58] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *ACM Symposium on the Theory of Computing (STOC)*, pages 448–456. ACM, 1983.

[59] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. In *ACM Symposium on the Theory of Computing (STOC)*, pages 246–251. ACM, 1983.

[60] J. Gajarskỳ, M. Lampis, and S. Ordyniak. Parameterized Algorithms for Modular-Width. In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013.

[61] A. Gajentaan and M. H. Overmars. On a class of $\wr(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

[62] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[63] R. Ganian. Thread graphs, linear rank-width and their algorithmic applications. In *International Workshop on Combinatorial Algorithms (IWOCA)*, volume 6460 of *Lecture Notes in Computer Science*, pages 38–42. Springer, 2010.

[64] C. Gavoille and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273(1):115–130, 2003.

[65] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Mathematics & Theoretical Computer Science*, 1(1):17–41, 1997.

[66] A. C. Giannopoulou, G. B. Mertzios, and R. Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical computer science*, 689:67–95, 2017.

[67] E. Gioan and C. Paul. Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics*, 160(6):708–733, 2012.

[68] M. C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science*, 11(03):423–443, 2000.

[69] M. Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

[70] F. Gurski and E. Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 1928 of *Lecture Notes in Computer Science*, pages 196–205. Springer, 2000.

[71] M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[72] E. Howorka. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.

[73] T. Husfeldt. Computing graph distances parameterized by treewidth and diameter. In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 63 of *Leibniz International Proceedings in Informatics*, pages 16:1–16:11. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[74] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 653–663. IEEE, 1998.

[75] Y. Iwata, T. Ogasawara, and N. Ohsaka. On the power of tree-depth for fully polynomial FPT algorithms. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96 of *Leibniz International Proceedings in Informatics*, pages 41:1–41:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[76] B. Jamison and S. Olariu. P-components and the homogeneous decomposition of graphs. *SIAM Journal on Discrete Mathematics*, 8(3):448–463, 1995.

[77] C. Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 1869(70):185–190, 1869.

[78] R. M. Karp and M. Sipser. Maximum matching in sparse random graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 364–375. IEEE, 1981.

[79] D. Kratsch and J. P. Spinrad. Between $\wr(nm)$ and $\wr(n^\alpha)$. *SIAM Journal on Computing*, 36(2):310–325, 2006.

[80] S. Kratsch and F. Nelles. Efficient and adaptive parameterized algorithms on modular decompositions. In *European Symposia on Algorithms (ESA)*, pages 55:1–55:15, 2018.

[81] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[82] J.-M. Lanlignel. *Autour de la décomposition en coupes*. PhD thesis, Université Montpellier 2, 2001.

[83] J. A. Makowsky and U. Rotics. On the clique-width of graphs with few P_4's. *International Journal of Foundations of Computer Science*, 10(03):329–348, 1999.

[84] G. B. Mertzios, A. Nichterlein, and R. Niedermeier. The power of linear-time data reduction for maximum matching. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 83 of *Leibniz International Proceedings in Informatics*, pages 46:1–46:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[85] S. Micali and V. V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27. IEEE, 1980.

[86] M. B. Novick. Fast parallel algorithms for the modular decomposition. Technical Report TR89-1016, Cornell University, 1989.

[87] S. Olariu. Weak bipolarizable graphs. *Discrete Mathematics*, 74(1-2):159–171, 1989.

[88] S.-i. Oum and P. D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[89] R. Puzis, Y. Elovici, P. Zilberman, S. Dolev, and U. Brandes. Topology manipulations for speeding betweenness centrality computation. *Journal of Complex Networks*, 3(1):84–112, Mar. 2015.

[90] Y. Rabinovich, J. A. Telle, and M. Vatshelle. Upper bounds on boolean-width with applications to exact algorithms. In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 8246 of *Lecture Notes in Computer Science*, pages 308–320. Springer, 2013.

[91] M. Rao. Clique-width of graphs defined by one-vertex extensions. *Discrete Mathematics*, 308(24):6157–6165, 2008.

[92] M. Rao. Solving some NP-complete problems using split decomposition. *Discrete Applied Mathematics*, 156(14):2768–2780, 2008.

[93] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[94] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *ACM Symposium on the Theory of Computing (STOC)*, pages 515–524. ACM, 2013.

[95] P. Scheffler. A linear algorithm for the pathwidth of trees. In *Topics in combinatorics and graph theory*, pages 613–620. Springer, 1990.

[96] M. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Univ. Paris Diderot (Paris 7), 2011.

[97] D. P. Sumner. Graphs indecomposable with respect to the X-join. *Discrete Mathematics*, 6(3):281–298, 1973.

[98] M. Tedder, D. G. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2008.

[99] V. Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 43 of *Leibniz International Proceedings in Informatics*, pages 16–28. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[100] V. Vassilevska Williams and R. R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 645–654. IEEE, 2010.

[101] M.-S. Yu and C.-H. Yang. An $O(n)$-time algorithm for maximum matching on cographs. *Information Processing Letter*, 47(2):89–93, 1993.

A Proof of Theorem 24

Our proof in this section involves a refinement of modules, that is called *p-connected components*. The notion of p-connectedness also generalizes connectivity in graphs. A graph $G = (V, E)$ is p-connected if and only if, for every bipartition (V_1, V_2) of V, there exists a path of length four with vertices in both V_1 and V_2. The p-connected components of a graph are its maximal induced subgraphs which are p-connected. Furthermore, a p-connected graph is termed *separable* if there exists a bipartition (V_1, V_2) of its vertex-set such that, for every crossing P_4, its two ends are in V_2 and its two internal vertices are in V_1. The latter bipartition (V_1, V_2) is called a separation, and if it exists then it is unique.

We need a strengthening of Theorem 1:

Theorem 27 ([76]). *For an arbitrary graph G exactly one of the following conditions is satisfied.*

1. *G is disconnected;*
2. *$\overline{G}$ is disconnected;*
3. *There is a unique proper separable p-connected component of G, with its separation being (V_1, V_2) such that every vertex not in this component is adjacent to every vertex of V_1 and nonadjacent to every vertex of V_2;*
4. *G is p-connected.*

If G or $\overline{G}$ is disconnected then it corresponds to a degenerate node in the modular decomposition tree. So we know how to handle with the two first cases. It remains to study the p-connected components of $(q, q-3)$-graphs.

For that, we need to introduce the class of p-trees:

Definition 3 ([6]). A graph G is a p-tree if one of the following conditions hold:

- the quotient graph G' of G is a P_4. Furthermore, G is obtained from G' by replacing one vertex by a cograph.
- the quotient graph G' of G is a spiked p-chain P_k, or its complement. Furthermore, G is obtained from G' by replacing any of x, y, v_1, v_k by a module inducing a cograph.
- the quotient graph G' of G is a spiked p-chain Q_k, or its complement. Furthermore, G is obtained from G' by replacing any of $v_1, v_k, z_2, z_3, \ldots, z_{k-5}$ by a module inducing a cograph.

We stress that the case where the quotient graph G' is a P_4, and so, of order $4 \leq 7 \leq q$ can be ignored in our analysis. Other characterizations for p-trees can be found in [5]. The above Definition 3 is more suitable to our needs.

Theorem 28 ([8]). *A p-connected component of a $(q, q-3)$-graph either contains less than q vertices, or is isomorphic to a prime spider, to a disc or to a p-tree.*

Finally, before we can prove Theorem 24, we need to further characterize the *separable* p-connected components. We use the following characterization of separable p-connected components.

Theorem 29 ([76]). *A p-connected graph G is separable if and only if its quotient graph is a split graph. Furthermore, its unique separation (V_1, V_2) is given by the union V_1 of the strong modules inducing the clique and the union V_2 of the strong modules inducing the stable set.*

We are now ready to prove Theorem 24.

Proof of Theorem 24. Suppose G and $\overline{G}$ are connected (otherwise we are done). By Theorem 27 there are two cases. First we assume G to be p-connected. By Theorem 28, G either contains less than q vertices, or is isomorphic to a prime spider, to a disc or to a p-tree. Furthermore, if G is a p-tree then according to Definition 3, the nontrivial modules can be characterized. So, we are done in this case. Otherwise, G is not p-connected. Let $V = V_1 \cup V_2 \cup V_3$ such that: $H = G[V_1 \cup V_2]$ is a separable p-component with separation (V_1, V_2), every vertex of V_3 is adjacent to every vertex of V_1 and nonadjacent to every vertex of V_2. Note that G' is obtained from the quotient graph

H' of H by possibly adding a vertex adjacent to all the strong modules in V_1. In particular, by Theorem 29 H' is a split graph, and so, G' is also a split graph. By Lemma 4, it implies that G' is either a prime spider, a spiked p-chain Q_k, a spiked p-chain $\overline{Q_k}$, or a graph with at most q vertices. Furthermore, if G' is a prime spider then so is H', and by Theorem 28 $H' = H$, hence G is a spider. Otherwise, G' is either a spiked p-chain Q_k or a spiked p-chain $\overline{Q_k}$. It implies that H is a p-tree. In particular, the nontrivial modules in H can be characterized according to Definition 3. The only nontrivial module of G that is not a nontrivial module of H (if any) contains V_3. Finally, since the module that contains V_3 has no neighbour among the modules in V_2, the corresponding vertex in the quotient can only be a z_i, for some i. So, we are also done in this case. $\square$