Fully polynomial FPT algorithms for some classes of bounded clique-width graphs

David Coudert, Guillaume Ducoffe, Alexandru Popa

Fully polynomial FPT algorithms for some classes of bounded clique-width graphs*

David Coudert† Guillaume Ducoffe†‡§ Alexandru Popa‡¶

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 $\mathcal{O}(k^4 \cdot n + m)$-time algorithm for computing a maximum matching where k is either the modular-width 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 graphs. Here 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.

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 [17, 26]. Some classes of perfect graphs, including distance-hereditary graphs, and so, trees, have bounded clique-width [51]. Furthermore, clique-width has many algorithmic applications. Many algorithmic schemes and metatheorems have been proposed for classes of bounded clique-width [25, 22, 34]. Perhaps the most famous one 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 [25]. 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 *finer-grained* complexity of polynomial graph problems when restricted to classes of bounded clique-width, that requires different tools than Courcelle's theorem.

Our starting point is the recent theory of "Hardness in P" that aims at better hierarchizing the complexity of polynomial-time solvable problems [77]. 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

*This work has been partially supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program "Investments for the Future" under reference ANR-11-LABX-0031-01.

†Université Côte d'Azur, Inria, CNRS, I3S, France

‡ICI – National Institute for Research and Development in Informatics, Romania

§The Research Institute of the University of Bucharest ICUB

¶University of Bucharest, Faculty of Mathematics and Computer Science

for one of these fundamental problems mentioned above. In their seminal work, Williams and Williams [78] prove that many important problems in graph theory are all equivalent under subcubic reductions. That is, if one of these problems admits a truly sub-cubic algorithms, then all of them do. Their results have extended and formalized prior work from, *e.g.*, [45, 60]. The list of such problems was further extended in [1, 14].

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.

Related work: Fully polynomial parameterized algorithms. FPT algorithms for polynomial-time solvable problems were first considered by Giannopoulou et al. [49]. 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 [49] proved that a matching of cardinality at least k in a graph can be computed in $\mathcal{O}(kn + k^3)$-time. We stress that Maximum Matching is a classical and intensively studied problem in computer science [31, 39, 40, 43, 59, 65, 64, 79]. The well known $\mathcal{O}(m\sqrt{n})$-time algorithm in [65] is essentially the best so far for Maximum Matching. Approximate solutions were proposed by Duan and Pettie [31].

More related to our work is the seminal paper of Abboud, Williams and Wang [3]. They obtained rather surprising results when using *treewidth*: another important graph parameter that intuitively measures the closeness of a graph to a tree [11]. Treewidth has tremendous applications in pure graph theory [72] and parameterized complexity [21]. Furthermore, improved algorithms have long been known for "hard" graph problems in P, such as Diameter and Maximum Matching, when restricted to trees [58]. However, it has been shown in [3] that under the Strong Exponential Time Hypothesis, for any $\varepsilon > 0$ there can be no $2^{o(k)} \cdot n^{2-\varepsilon}$-time algorithm for computing the diameter of graphs with treewidth at most k. 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 [18]. However, clique-width cannot be *polynomially* upper-bounded by the treewidth [18]. Thus, the hardness results from [3] 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.

On a more positive side, the authors in [3] show that Radius and Diameter can be solved in $2^{\mathcal{O}(k \log k)} \cdot n^{1+\mathcal{O}(1)}$-time, where k is treewidth. Husfeldt [55] shows that the eccentricity of every vertex in an undirected graph on n vertices can be computed in time $n \cdot \exp[\mathcal{O}(k \log d)]$, where k and d are the treewidth and the diameter of the graph, respectively. More recently, a tour de force was achieved by Fomin et al. [38] who were the first to design parameterized algorithms with *polynomial dependency* on the treewidth, for Maximum Matching and Maximum Flow. Furthermore they proved that for graphs with treewidth at most k, a tree decomposition of width $\mathcal{O}(k^2)$ can be computed in $\mathcal{O}(k^7 \cdot n \log n)$-time. We observe that their algorithm for Maximum Matching is *randomized*, whereas ours are deterministic.

We are not aware of the study of another parameter than treewidth for polynomial graph problems. However, some authors choose a different approach where they study the parameterization of a fixed graph problem for a range of graph invariants [9, 37, 64]. As examples, *clique-width* is part of the graph invariants used in the parameterized study of Triangle Listing [9]. Nonetheless, clique-width is not the main focus in [9]. Mertzios, Nichterlein and Niedermeier [64] propose algorithms for Maximum Matching that run in time $\mathcal{O}(k^{\mathcal{O}(1)} \cdot (n + m))$, for several parameters such as feedback vertex set or feedback edge set. Moreover, the authors in [64] suggest that Maximum Matching may become the "drosophila" of the study of the FPT algorithms in P. We advance in this research direction.

1.1 Our results.

In this paper 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* [26], *split-width* [71], *neighbourhood diversity* [61] and P_4-*sparseness* [7]. 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 3.1 and 3.2). The latter complements the results obtained for Triangle Listing in [9]. However many hardness results for *distance problems* when using treewidth are proved to also hold when using clique-width (Theorems 4.1, 4.2 and 4.3). 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 algorithm for computing the clique-width of a graph remains a challenging open problem [16]. 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	Our result
Diameter, Eccentricities	$\mathcal{O}(mw(G)^3 + n + m)$ $\mathcal{O}(sw(G)^2 \cdot n + m)$ $\mathcal{O}(q(G)^3 + n + m)$
Betweenness Centrality	$\mathcal{O}(mw(G)^2 \cdot n + m)$ $\mathcal{O}(sw(G)^2 \cdot n + m)$ $\mathcal{O}(nd(G)^3 + n + m)$
Hyperbolicity	$\mathcal{O}(mw(G)^3 \cdot n + m)$ $\mathcal{O}(nd(G)^4 + n + m)$ $\mathcal{O}(sw(G)^3 \cdot n + m)$ $\mathcal{O}(q(G)^3 \cdot n + m)$
Maximum Matching	$\mathcal{O}(mw(G)^4 \cdot n + m)$ $\mathcal{O}(q(G)^4 \cdot n + m)$
Triangle Counting, Girth	$\mathcal{O}(cw(G)^2 \cdot (n + m))$

Table 1: Summary of positive results.

Graph parameters and decompositions considered. Let us describe the parameters considered in this work as follows. The following is 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* [50].

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 4.4, 4.5 and 4.6, respectively). Although clique-width is also related to some treelike representations of graphs [24], 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 *modular decomposition* of a graph is a hierarchical decomposition that partitions the vertices of the graph with respect to the modules [54]. Split decomposition is often presented as a refinement of modular decomposition [50]. We formalize the relationship between the two in Lemma 4.3, 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 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 $\mathcal{O}(k^{\mathcal{O}(1)} \cdot n + m)$ when parameterized by the *split-width* (maximum order of a prime subgraph in the split decomposition), to $\mathcal{O}(k^{\mathcal{O}(1)} + n + m)$-time when parameterized by the *modular-width* (maximum order of a prime subgraph in the modular decomposition). See Theorem 4.7.

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). New kernelization results are obtained for Hyperbolicity and Betweenness Centrality when parameterized by the neighbourhood diversity (Theorems 4.9 and 4.10). 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, introduced by Babel and Olariu in [6], 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, $t \leq q-3$, the class of (q,t)-graphs has bounded clique-width [63]. We so define the P_4-sparseness of a graph G, denoted $q(G)$, 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 has been extensively studied and characterized in the literature [4, 6, 7, 5, 57]. We take advantage of these existing characterizations in order to generalize our algorithms with modular decomposition to $\mathcal{O}(q(G)^{\mathcal{O}(1)} \cdot n + m)$-time algorithms (Theorems 4.11 and 4.12).

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 [14]. 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 [4, 6, 7, 5, 57] 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.

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 [5, 47, 50, 44, 71]. However, we think we are the first to study clique-width and its upper-bounds for polynomial problems. 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 [79], distance-hereditary graphs [29, 30], P_4-tidy graphs [40], 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 k, k being any fixed constant. For instance, the algorithm that is proposed in [30] 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 [50]. 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 [64] that MAXIMUM MATCHING may become the "drosophila" of the study of the FPT algorithms in P. Precisely, we propose $\mathcal{O}(k^4 \cdot n + m)$-time algorithms for MAXIMUM MATCHING when parameterized either by modular-width or by the P_4-sparseness of the graph (Theorems 5.2 and 5.4). The latter subsumes many algorithms that have been obtained for specific subclasses [40, 79].

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 [10]. In an unpublished paper [66], Novick followed a similar approach and, based on an integer programming formulation, he obtained an $\mathcal{O}(k^{\mathcal{O}(k^3)}n+m)$-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 of modular-width at most k we only need to consider augmenting paths of length $\mathcal{O}(k)$. Then, our second contribution is an efficient way to compute such paths. For that, we design a new type of characteristic graph of size $\mathcal{O}(k^4)$. The same as the classical quotient graph 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.

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. Indeed, we need to go deeper in the p-connectedness theory of [7] in order to better characterize the nontrivial modules in the graphs (Theorem 5.3). 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 [59, 79], that is of independent interest.

We suspect that our algorithm with modular decomposition can be used as a subroutine in order to solve Maximum Matching in linear-time for bounded split-width graphs. However, this is left for future work.

1.2 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 3.2). 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 $\mathcal{O}(k^2 L)$ 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 4.1, none of these problems above can be solved in time $2^{o(k)}n^{2-\varepsilon}$, for any $\varepsilon > 0$, when parameterized by the clique-width (Theorems 4.1—4.3). 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 [56]. In summary, we take advantage of recent hardness results obtained for *bounded-degree* graphs [35]. Clique-width and treewidth can only differ by a constant-factor in the class of bounded-degree graphs [22, 53]. 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.

In Section 4.2 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 4.3 we apply the results from Section 4.2 to modular-width. First, since $sw(G) \leq mw(G) + 1$ for any graph G, 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 only to process the quotient graph of G. We thus obtain algorithms that run in $\mathcal{O}(mw(G)^{\mathcal{O}(1)}+n+m)$-time, or $\mathcal{O}(nd(G)^{\mathcal{O}(1)}+n+m)$-time, for all these problems. In Section 4.4 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 2.4.

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

Finally, in Section 6 we discuss applications to other graph classes. Due to lack of space, most of the proofs are sketched or omitted. They can be found in our technical report [20].

2 Preliminaries

We use standard graph terminology from [12, 28]. Graphs in this study are finite, simple (hence without loops or multiple edges) and unweighted – unless stated otherwise. Furthermore we assume that graphs are encoded as adjacency lists. 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 i and the set of vertices labeled j, where $i \neq j$, denoted $\eta(i, j)$;

4. Renaming label i to label j, denoted $\rho(i, j)$.

The *clique-width* of G, denoted by $cw(G)$, is the minimum k such that, for some labeling ℓ, the labeled graph $\langle G, \ell \rangle$ admits a k-expression [23]. We refer to [25] and the references cited therein for a survey of the many applications of clique-width in the field of parameterized complexity. Computing the clique-width of a given graph is NP-hard [36]. However, on a more positive side the graphs with clique-width two are exactly the cographs and they can be recognized in linear-time [17, 26]. Clique-width three graphs can also be recognized in polynomial-time [16]. The parameterized complexity of computing the clique-width is open. 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 $M \subseteq V(G)$ 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 $\mathcal{M}(G)$ be the family of all inclusion wise maximal strong modules of G that 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 2.1. ([46]) *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 2.1 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 $G = (V, E)$ can be computed in linear-time [76]. See Fig. 1 for an example.

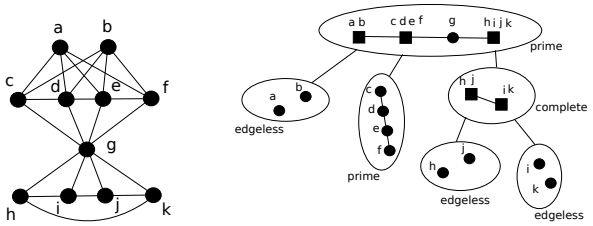

Figure 1: A graph and its modular decomposition.

Furthermore, by Theorem 2.1 the subgraphs from the modular decomposition are either edgeless, complete, or prime for modular decomposition. The *modular-width* [1] of G, denoted by $mw(G)$, is the minimum $k \geq 2$ such that any prime subgraph in the modular decomposition has order (number of vertices) at most k. The relationship between clique-width and modular-width is as follows.

LEMMA 2.1. ([25]) *For every $G = (V, E)$, we have $cw(G) \leq mw(G)$, and a $mw(G)$-expression defining G can be constructed in linear-time.*

We refer to [54] 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 [75]. Cographs enjoy many algorithmic properties, including a linear-time algorithm for MAXIMUM MATCHING [79]. Furthermore, in [44] Gajarskỳ, Lampis and Ordyniak prove that for some W-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 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 bipartition of their vertex-set is a split. They are exactly the complete graphs and the stars [27]. 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,

[1] This term has another meaning in [70]. We rather follow the terminology from [26].

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 [15]. The *split-width* of G, denoted by $sw(G)$, 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. 2 for an illustration.

LEMMA 2.2. ([71]) *For every $G = (V, E)$, we have $cw(G) \leq 2 \cdot sw(G) + 1$, and a $(2 \cdot sw(G) + 1)$-expression defining G can be constructed in linear-time.*

We refer to [47, 50, 71] for some algorithmic applications of split decomposition. In particular, graphs with split-width at most two are exactly the distance-hereditary graphs [8]. Linear-time algorithms for solving DIAMETER and MAXIMUM MATCHING for distance-hereditary graphs are presented in [30, 29].

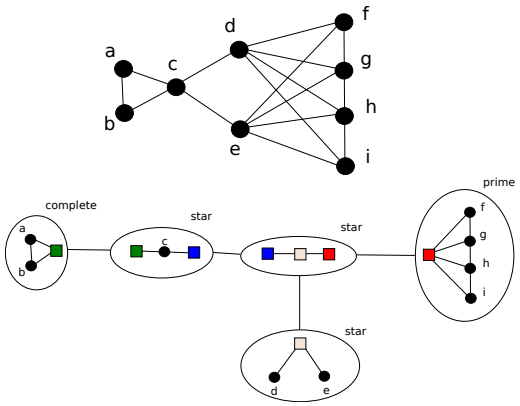

Figure 2: A graph and its split decomposition.

We stress that split decomposition can be seen as a refinement of modular decomposition. Indeed, if M is a module of G 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 [6]. The *P_4-sparseness* of G, denoted by $q(G)$, is the minimum $q \geq 7$ such that G is a $(q, q-3)$-graph.

LEMMA 2.3. ([63]) *For every $q \geq 7$, every $(q, q-3)$-graph has clique-width at most q, and a q-expression defining it can be computed in linear-time.*

The algorithmic properties of several subclasses of $(q, q-3)$-graphs have been considered in the literature. We refer to [7] for a survey. In particular, 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 obtained from the join $G[K] \oplus G[R]$, between a complete graph $G[K]$ and an arbitrary $G[R]$, by adding a new set S of vertices, $|S| = |K|$, and adding edges between S and K that induce 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*. If furthermore $|R| \leq 1$ then we call G a *prime spider*. A *spiked p-chain* P_k is a supergraph of the path P_k, possibly with the additional vertices x, y such that: $N(x) = \{v_2, v_3\}$ and $N(y) = \{v_{k-2}, v_{k-1}\}$. Note that one or both of x and y may be missing. A *spiked p-chain* $\overline{P_k}$ is the complement of a spiked p-chain P_k. Finally, let Q_k be the graph with vertex-set $\{v_1, v_2, \ldots, v_k\}$, $k \geq 6$ such that, for every $i \geq 1$, $N_{Q_k}(v_{2i-1}) = \{v_{2j} \mid j \leq i,\ j \neq i-1\}$ and $N_{Q_k}(v_{2i}) = \{v_{2j} \mid j \neq i\} \cup \{v_{2j-1} \mid j \geq i,\ j \neq i+1\}$. 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: $N(z_{2i-1}) = \{v_{2j} \mid j \in [1; i]\} \cup \{z_{2j} \mid j \in [1; i-1]\}$ and $N(z_{2i}) = \{v_{2j-1} \mid j \in [1; i+1]\} \cup \{z_{2j-1} \mid j \in [2; i]\}$. Any of the vertices z_i can be missing. A *spiked p-chain* $\overline{Q_k}$ is the complement of a spiked p-chain Q_k.

A *prime p-tree* 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 [63].

LEMMA 2.4. ([5, 63]) *Let $G = (V, E)$, $q \geq 7$, be a connected $(q, q-3)$-graph 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 $|V(G')| \leq q$.*

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. Our basic cycle problem is TRIANGLE DETECTION: is a given $G = (V, E)$ triangle-free ? Positive results are obtained for the two variations TRIANGLE COUNTING (number of triangles in G) and GIRTH

(minimum size of a cycle in G). In [78], the three above problems are proved to be subcubic equivalent when restricted to combinatorial algorithms.

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 [78]. 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. 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 3.1. *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.*

Sketch 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 [26]. Then, we proceed by dynamic programming on the irredundant k-expression.

More precisely, let $\langle G, \ell \rangle$ be a labeled graph, $\ell : V(G) \to \{1, \ldots, k\}$. 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 $p, q \in \{1, \ldots, k\}$, 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. 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, it suffices to observe that, after any operation, we can update the values $m_{p,q}$ and $n_{p,q}$ in $\mathcal{O}(k^2)$-time. □

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 3.2 gives the first known polynomial parameterized algorithm for Girth.

Theorem 3.2. *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.*

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 [68]. 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 3.1 and 3.2 with Lemmas 2.1, 2.2 and 2.3 we obtain the following result.

Corollary 3.1. *For every $G = (V, E)$, Triangle Counting and Girth can be solved in $\mathcal{O}(k^2 \cdot (n+m))$-time, for every $k \in \{mw(G), sw(G), q(G)\}$.*

4 Separability results using distance problems

We prove separability results between clique-width and the upper-bounds for clique-width presented in Sec. 2. More precisely, we consider the following problems.

Given $G = (V, E)$ connected, the *eccentricity* of a given vertex v is defined as $ecc_G(v) = \max_{u \in V} dist_G(u, v)$. The *diameter* of G is defined as $diam(G) = \max_v ecc_G(v)$. In the Diameter problem, resp. in the Eccentricities problem, we ask for the diameter of G, resp. for the eccentricities of all the vertices in G. Hardness results for Diameter have been proved, *e.g.*, in [73, 1, 14, 3, 35].

Gromov hyperbolicity is a measure of how close (locally) the shortest-path metric of a graph G is to a tree metric [52]. Formally, the hyperbolicity of G, denoted $\delta(G)$, is the smallest δ such that there exists a tree embedding with additive distortion at most δ for every four vertices of G. We refer to [32] for a survey on the applications of Gromov hyperbolicity in computer science. Hardness results for Hyperbolicity have been proved in [14, 19, 41]. Some fully polynomial parameterized algorithms, with a different range of

parameters than the one considered in this work, have been designed in [37].

Finally, the betweenness centrality of a given vertex v is equal to the sum, over all pairs $s, t \in V(G) \setminus \{v\}$, of the number of shortest st-paths passing by v divided by the total number of shortest st-paths in G [42]. In the Betweenness Centrality problem, we ask for the betweenness centrality of all the vertices in G. See [1, 14, 35] for hardness results on this problem.

On the negative side, we show in Section 4.1 that we cannot solve these 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 4.2, 4.3 and 4.4, when parameterized by either the modular-width, the split-width or the P_4-sparseness.

4.1 Hardness results for clique-width. The goal in this section is to prove that we cannot solve the distance problems considered in this work in time $2^{o(cw)}n^{2-\varepsilon}$, for any $\varepsilon > 0$ (Theorems 4.1—4.3). 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 cannot be solved in $\mathcal{O}^*(2^{c \cdot n})$-time, for any $c < 1$ [56]. 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. The *width* of a tree decomposition is the size of a largest bag minus one. Finally, the *treewidth* of a graph G, denoted by $tw(G)$, is the least possible width over its tree decompositions.

Several hardness results have already been obtained for treewidth [3]. However, $cw(G) \leq 2^{tw(G)}$ for general graphs [18], that does not help us to derive our lower-bounds. Roughly, we use relationships between treewidth and clique-width in some graph classes (*i.e.*, bounded-degree graphs [22, 53]) in order to transpose the hardness results for treewidth into hardness results for clique-width. Namely:

Lemma 4.1. ([22, 53]) *If G has maximum degree at most d (with $d \geq 1$), we have:*

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

Our reductions in what follows are based on Lemma 4.1, and on previous hardness results for bounded treewidth graphs and bounded-degree graphs [3, 35].

Theorem 4.1. *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 clique-width at most k, for any $\varepsilon > 0$.*

Sketch proof. In [3], they 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, $tw(G) = \mathcal{O}(|C|) = \mathcal{O}(\log n)$ [3].

Then, we use the generic construction of [35] in order to transform G into a bounded-degree graph G'. We prove that graphs with treewidth $\mathcal{O}(\log n)$ can be generated with this construction[2]. 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 [35]. Our main contribution is to upper-bound the clique-width and the treewidth of their construction.

Theorem 4.2. *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 clique-width at most k, for any $\varepsilon > 0$.*

Our last reduction for Hyperbolicity is inspired from the one presented in [14]. However, the authors in [14] 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.

[2] Our construction has fewer degrees of freedom than the construction presented in [35].

THEOREM 4.3. *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* $\varepsilon > 0$.

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, 55]).

4.2 Parameterized algorithms with split decomposition. We 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 4.3. Roughly, we show that in order to solve the above problems, 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 tree T where the nodes are in bijective correspondence with the subgraphs of the split decomposition of G, and the edges of T are in bijective correspondence 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. 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 [15].

We illustrate our approach with the ECCENTRICITIES problem. The following lemma is easy to prove:

LEMMA 4.2. *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:*

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

THEOREM 4.4. *For every* $G = (V, E)$*,* ECCENTRICITIES *can be solved in* $\mathcal{O}(sw(G)^2 \cdot n + m)$*-time. In particular,* DIAMETER *can be solved in* $\mathcal{O}(sw(G)^2 \cdot n + m)$*-time.*

Proof. Let T be a split decomposition tree of G, with its nodes being in bijective correspondence with the split components $C_1, C_2, \ldots, C_k$. It can be computed in linear-time [15]. 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, where $V(C_i) \subseteq A_i$. 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. We observe that T_i is a split decomposition tree of G_{A_i}, $T \setminus T_i$ 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 $\mathcal{O}(|V(C_i)|^3) = \mathcal{O}(sw(G)^2 \cdot |V(C_i)|)$-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 $V(C_{i_t}) \subseteq A_{i_t}$. 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 4.2 for every split (A_{i_t}, B_{i_t}) we get:

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

  We distinguish between three cases.

  1. If C_i is complete, then we need to compute $x_i \in V(C_i)$ 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 compute 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. Else, $|V(C_i)| \leq sw(G)$, and so, the eccentricities can be computed in $\mathcal{O}(|V(C_i)||E(C_i)|) = \mathcal{O}(sw(G)^2 \cdot |V(C_i)|)$-time.

Overall, this step takes total time $\mathcal{O}(sw(G)^2 \cdot \sum_i |V(C_i)|) = \mathcal{O}(sw(G)^2 \cdot n)$. 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 4.2 we have, for every $v_i \in V(G_i)$:

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

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 $V(C_{j_t}) \subseteq A_{j_t}$. 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 $V(C_{p(i)}) \subseteq A_{p(i)}$. So, there has 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 4.2 it comes that for all $0 \leq t \leq k$ we have:

$$ecc_{G_{B_{i_t}}}(a_{i_t}) = \max_{v \in V(C_{p(i)}) \setminus \{a_{i_t}\}} dist_{C_{p(i)}}(a_{i_t}, v) + e'(v).$$

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

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

By using the same approach as for Theorem 4.4, *i.e.*, dynamic programming on the split decomposition tree, we also prove the following two results.

THEOREM 4.5. *For every $G = (V, E)$,* HYPERBOLICITY *can be solved in $\mathcal{O}(sw(G)^3 \cdot n + m)$-time.*

THEOREM 4.6. *For every $G = (V, E)$,* BETWEENNESS CENTRALITY *can be solved in $\mathcal{O}(sw(G)^2 \cdot n + m)$-time.*

4.3 Kernelization methods with modular decomposition. The purpose of this 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. We thus obtain algorithms that run in $\mathcal{O}(mw(G)^{\mathcal{O}(1)} + n + m)$-time. See [64] for an extended discussion on the use of *Kernelization* for graph problems in P.

We start with the following lemma:

LEMMA 4.3. (FOLKLORE) *For every $G = (V, E)$ we have $sw(G) \leq mw(G) + 1$.*

The proof of Lemma 4.3 is constructive. It shows how to compute from the modular decomposition of G a partial split decomposition[3] where every non degenerate subgraph has order at most $mw(G) + 1$. Furthermore, this above transformation can be computed in linear time.

COROLLARY 4.1. *For every $G = (V, E)$ we can solve:*

- ECCENTRICITIES *and* DIAMETER *in $\mathcal{O}(mw(G)^2 \cdot n + m)$-time;*
- HYPERBOLICITY *in $\mathcal{O}(mw(G)^3 \cdot n + m)$-time;*
- BETWEENNESS CENTRALITY *in $\mathcal{O}(mw(G)^2 \cdot n + m)$-time.*

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

THEOREM 4.7. *For every $G = (V, E)$,* ECCENTRICITIES *(and so,* DIAMETER*) can be solved in time $\mathcal{O}(mw(G)^3 + n + m)$.*

Proof. W.l.o.g., G is connected. By Lemma 4.3 there is a (partial) split decomposition, obtained from the modular decomposition of G, with the following properties. There exists a modular partition $M_1, M_2, \ldots, M_k$ of G such that:

- 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

[3] This split decomposition is "partial", in the sense that there may be non degenerate split components that can be further decomposed.

degenerate (and so, $diam(G') \leq 2$) or $k \leq mw(G)+1$. We can also observe in this situation that if we root the corresponding split decomposition tree 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 4.3 $G = G[M_1] \oplus G[M_2]$. 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 $\mathcal{O}(mw(G)^3)$-time otherwise. Applying the algorithmic scheme of Theorem 4.4, 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. □

Next, we consider Hyperbolicity. It is proved in [74] that, for every $G = (V, E)$ with quotient graph G', $\delta(G') \leq \delta(G) \leq \max\{\delta(G'), 1\}$. The latter immediately implies the following result:

Theorem 4.8. *For every $G = (V, E)$, we can decide whether $\delta(G) > 1$, and if so, compute $\delta(G)$, in $\mathcal{O}(mw(G)^4 + n + m)$-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 4.4. ([19]) *For every $G = (V, E)$ 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 $\mathcal{O}(mw(G)^{\mathcal{O}(1)} + n + m)$-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 distance problems we consider. 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, sometimes denoted by $nd(G)$ [61]. Observe that every set of pairwise twins is a module of G. Hence, $mw(G) \leq nd(G)$.

Theorem 4.9. *For every $G = (V, E)$, Hyperbolicity can be solved in $\mathcal{O}(nd(G)^4 + n + m)$-time.*

Proof. Let $V_1, V_2, \ldots, V_k$, $k = nd(G)$, partition the vertex-set V in twin classes. The partition can be computed in linear-time [61]. Furthermore, since it is a modular partition, we can compute a (partial) split decomposition as described in Lemma 4.3. 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 4.5, 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 (*i.e.*, vertices with a neighbourhood inducing a complete subgraph)[4]. This can be done in $\mathcal{O}(|V(C_j)|)$-time if C_j is a star or a complete graph, and in $\mathcal{O}(nd(G)^4)$-time if $C_j = G'$ (*e.g.*, see [13]). Hence, we can solve Hyperbolicity for G in $\mathcal{O}(nd(G)^4 + n + m)$-time. □

In [37], the authors propose an $\mathcal{O}(2^{\mathcal{O}(k)} + n + m)$-time algorithm for computing Hyperbolicity with k being the vertex-cover of the graph. Their algorithm is pretty similar to Theorem 4.9. This is no coincidence since every graph with vertex-cover at most k has neighbourhood diversity at most $2^{\mathcal{O}(k)}$ [61].

Finally, the following was proved implicitly in [69].

Theorem 4.10. ([69]) *For every $G = (V, E)$, Betweenness Centrality can be solved in $\mathcal{O}(nd(G)^3 + n + m)$-time.*

4.4 Applications to graphs with few P_4's. Before ending Sec. 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 2.4).

Theorem 4.11. *For every $G = (V, E)$, Eccentricities can be solved in $\mathcal{O}(q(G)^3 + n + m)$-time.*

Theorem 4.12. *For every $G = (V, E)$, Hyperbolicity can be solved in $\mathcal{O}(q(G)^3 \cdot n + m)$-time.*

[4] Simply put, we need to compute the split marker vertices that are simplicial in order to decide whether $\delta(G) > \max_j \delta(C_j)$.

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 4.6. However, this would require to find ad-hoc methods for every graph family in Lemma 2.4. 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 cardinality. Maximum Matching can be solved in polynomial time with Edmond's algorithm [33]. Micali and Vazirani [65] show how to implement Edmond's algorithm in time $\mathcal{O}(m\sqrt{n})$. In [64], Mertzios, Nichterlein and Niedermeier design some new algorithms to solve Maximum Matching, that run in $\mathcal{O}(k^{\mathcal{O}(1)} \cdot (n+m))$-time for various graph parameters k. 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 [40, 79].

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 5.1. (Berge, [10]) *A matching F in $G = (V, E)$ 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 $\mathcal{O}(mw(G))$) 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 5.1. *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 $G[M]$. 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. □

From now on, let us assume each module induces a matching. Then, we need to upper-bound the number of edges in an augmenting path.

Lemma 5.2. *Let $G = (V, E)$ be a graph such that every module $M \in \mathcal{M}(G)$ induces a matching. If $F \subseteq E$ is a non maximum matching of G then there is 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 $\{i \mid x_{2i-1}, x_{2i} \in M\} \neq \emptyset$, and $M' \subseteq N_G(M)$, then we have $\{i \mid x_{2i-1}, x_{2i} \in M'\} = \emptyset$;
- $|\{i \mid x_{2i}, x_{2i+1} \in M\}| \leq 1$;
- $|\{i \mid x_{2i-1} \notin M,\ x_{2i} \in M\}| \leq 1$;
- $|\{i \mid x_{2i} \notin M,\ x_{2i+1} \in M\}| \leq 2$;
  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 $\mathcal{O}(|\mathcal{M}(G)|)$.

Based on Lemmas 5.1 and 5.2, 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 is 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 replacing every $M \in \mathcal{M}(G)$ with at most one edge and at most two isolated vertices. More generally, we can upper-bound the size of the witness subgraph, as follows.

LEMMA 5.3. *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 $\mathcal{O}(|E(G')|) = \mathcal{O}(mw(G)^2)$.*

Our algorithm is based on the correspondence between F-augmenting paths in G and F'-augmenting paths in G'_F, that we prove next. The following Lemma 5.4 is the key technical step of the algorithm.

LEMMA 5.4. *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.*

Sketch 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 5.2.

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 5.2 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 5.2 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. However, we observe that in this situation, we always have $M_{2i-1} = M_{2i}$. 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 $\{w_i, x_i\}, \{y_i, z_i\} \in F'$. Roughly, we insert alternating subpaths in the sequence $\mathcal{S}_P$, using edges $\{w_i, x_i\}$ and $\{y_i, z_i\}$, 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. For that, we need to consider eight possible configurations, that are not detailed here due to lack of space.□

We can now state the main result in this subsection.

THEOREM 5.2. *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 5.1 (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 2.1 G' is either prime for modular decomposition or a complete graph. First, 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 $\mathcal{O}(|E(G')|) = \mathcal{O}(mw(G)^2)$. 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 $2\ell - 1$.

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 $\mathcal{O}(|E(G')|^2) = \mathcal{O}(mw(G)^4)$ by Lemma 5.3. Furthermore by Theorem 5.1 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 5.4 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 [43], that is in $\mathcal{O}(mw(G)^4)$. 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 $\mathcal{O}(mw(G)^4 \cdot t_{\max})$.

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 $\mathcal{O}(n)$ [71]. Furthermore, a maximum matching of G also has cardinality $\mathcal{O}(n)$. Therefore, the total running time is in $\mathcal{O}(mw(G)^4 \cdot n)$ if the modular decomposition of G is given. The latter decomposition can be precomputed in $\mathcal{O}(n + m)$-time [76]. □

5.2 More structure: $(q, q-3)$-graphs. The second main result in Section 5 is an $\mathcal{O}(q(G)^4 \cdot n + m)$-time algorithm for Maximum Matching (Theorem 5.4). Our algorithm for $(q, q-3)$-graphs reuses the algorithm described in Theorem 5.2 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 2.4. 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 2.4. The latter generalizes similar structure theorems that have been obtained for some specific subclasses [48].

Theorem 5.3. *For any $(q, q-3)$-graph G, $q \geq 7$, and its quotient graph G', one of the following holds.*

1. *G is disconnected;*
2. *$\overline{G}$ is disconnected;*
3. *G is a disc (and so, $G = G'$ is prime);*
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, $v \in V(G')$ can represent a non trivial module only if $v \in \{v_1, v_k, x, y\}$;*
6. *G' is a spiked p-chain Q_k, or a spiked p-chain $\overline{Q_k}$. Furthermore, $v \in V(G')$ can represent a non trivial module only if $v \in \{v_1, v_k, z_2, z_3, \ldots, z_{k-5}\}$;*
7. *$|V(G')| \leq q$.*

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

Lemma 5.5. *For every disc $G = (V, E)$, a maximum matching can be computed in linear-time.*

Lemma 5.6. *If $G = (S \cup K \cup R, E)$ is a spider then there is a maximum matching of G composed of: a perfect matching between K and S; and a maximum matching of $G[R]$.*

Finally, 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 5.7).

3. Apply a technique known as "SPLIT and MATCH" [79] 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 5.8.

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 [59].

Lemma 5.7. *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 5.1, 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 $\{u, v\}$ [59]. 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 [59]: 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 $\{x, y\} \in F_M$ 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 [79].

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

The "SPLIT and MATCH" has been applied to compute a maximum matching in linear-time for cographs and some of its generalizations [39, 40, 79]. Our Theorem 5.2 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 5.8. *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 [79] 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 5.1, a maximum matching for G^* is a maximum matching for G. □

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

Proposition 5.1. *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.*

Sketch proof. Due to lack of space, we only prove the case where G is a spiked p-chain Q_k. For every $v \in V(G')$, let $M_v \in \mathcal{M}(G)$ be the corresponding module. 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 $G_i = G[V_i \cup U_{i-1}]$,

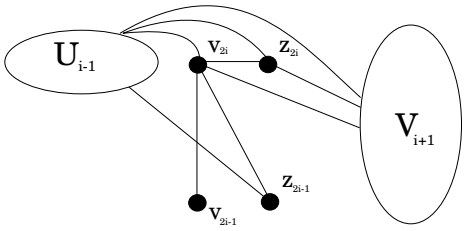

Figure 3: Schematic view of graph G_i.

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. 3.

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 5.2 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 5.3, $v_{2i} \in V(G)$, hence $M_{v_{2i-1}}$ is a pending module of G_i. Thus, we can apply the reduction rule of Lemma 5.7. 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 5.8 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 5.8 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 5.7. 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. □

Main result.

THEOREM 5.4. *For every $G = (V, E)$,* MAXIMUM MATCHING *can be solved in $\mathcal{O}(q(G)^4 \cdot n + m)$-time.*

Proof. We generalize the algorithm for Theorem 5.2. 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 $q(G)$ vertices then we apply the same techniques as for Theorem 5.2 in order to compute a maximum matching $F_{\max}$ for G. It takes constant-time if G' is a stable, $\mathcal{O}(q(G)^4 \cdot (|F_{\max}| - |F^*|))$-time if G' is prime and $\mathcal{O}(|V(G')| + (|F_{\max}| - |F^*|))$-time if G' is a complete graph. Otherwise by Theorem 5.3 the following cases need to be considered.

- Suppose G is a disc. In particular, $G = G'$. By Lemma 5.5, 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 5.6, 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 5.1, 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 $\mathcal{O}(n)$ [71]. Summing the size of all the subgraphs in the modular decomposition of G amounts to $\mathcal{O}(n + m)$ [71]. Furthermore, a maximum matching of G also has cardinality $\mathcal{O}(n)$. Therefore, the total running time is in $\mathcal{O}(q(G)^4 \cdot n + m)$ if the modular decomposition of G is given. The latter decomposition can be precomputed in $\mathcal{O}(n + m)$-time [76]. □

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 [67]. 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 [62]. 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 [70].

THEOREM 6.1. ([70]) *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 6.1. *Let $G = (V, E)$ 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 4.3 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. □

We now revisit the algorithmic scheme that was presented earlier for ECCENTRICITIES (Theorem 4.4).

PROPOSITION 6.1. *For every $G = (V, E)$ 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 4.7 the problem reduces in linear-time to solve ECCENTRICITIES for G'. We compute the split-decomposition of G'. It takes linear-time [15]. By Lemma 6.1 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 4.4 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 6.1 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. □

References

[1] A. Abboud, F. Grandoni, and V. Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *SODA'15*, pp 1681–1697.

[2] A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS'14*, pp 434–443.

[3] A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA'16*, pp 377–391.

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

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

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

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

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

[9] M. Bentert, T. Fluschnik, A. Nichterlein, and R. Niedermeier. Parameterized Aspects of Triangle Enumeration. In *FCT'17*, pp 96–110.

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

[11] H. Bodlaender. Treewidth: Characterizations, Applications, and Computations. In *WG'06*, pp 1–14.

[12] J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.

[13] M. Borassi, D. Coudert, P. Crescenzi, and A. Marino. On computing the hyperbolicity of real-world graphs. In *ESA'15*, pp 215–226.

[14] 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.

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

[16] D. Corneil, M. Habib, J.-M. Lanlignel, B. Reed, and U. Rotics. Polynomial Time Recognition of Clique-Width ≤ 3 Graphs. In *LATIN'00*, pp 126–134.

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

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

[19] 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.

[20] D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. Technical Report, https://hal.archives-ouvertes.fr/hal-01562413, 2017.

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

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

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

[24] 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.

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

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

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

[28] R. Diestel. *Graph Theory*. Grad. Texts in Math. Springer, 2010. 4*th* edition.

[29] F. Dragan. On greedy matching ordering and greedy matchable graphs. In *WG'97*, pp 184–198.

[30] 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.

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

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

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

[34] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *WG'11*, pp 117–128.

[35] J. Evald and S. Dahlgaard. Tight Hardness Results for Distance and Centrality Problems in Constant Degree Graphs. Technical Report arXiv:1609.08403, ArXiv, 2016.

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

[37] T. Fluschnik, C. Komusiewicz, G. Mertzios, A. Nichterlein, R. Niedermeier, and N. Talmon. When can Graph Hyperbolicity be computed in Linear Time? In *WADS'17*, pp 397–408.

[38] F. Fomin, D. Lokshtanov, M. Pilipczuk, S. Saurabh, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. In *SODA'17*, pp 1419–1432.

[39] J.-L. Fouquet, V. Giakoumakis, and J.-M. Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International J. of Foundations of Computer Science*, 10(04):513–533, 1999.

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

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

[42] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[43] H. Gabow and R. Tarjan. A linear-time algorithm for a special case of disjoint set union. In *STOC'83*, pp

246–251.

[44] J. Gajarskỳ, M. Lampis, and S. Ordyniak. Parameterized Algorithms for Modular-Width. In *IPEC'13*, pp 163–176.

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

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

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

[48] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Mathematics and Theoretical Computer Science*, 1, 1997.

[49] 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*, 2017. In Press.

[50] 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.

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

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

[53] F. Gurski and E. Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *WG'00*, pp 196–205.

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

[55] T. Husfeldt. Computing graph distances parameterized by treewidth and diameter. In *IPEC'16*, volume 63 pp 16:1–16:11.

[56] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *FOCS'98*, pp 653–662.

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

[58] C. Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.

[59] R. Karp and M. Sipser. Maximum matching in sparse random graphs. In *FOCS'81*, pp 364–375.

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

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

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

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

[64] G. Mertzios, A. Nichterlein, and R. Niedermeier. The power of data reduction for matching. In *MFCS'17*, to appear.

[65] S. Micali and V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *FOCS'80*, pp 17–27.

[66] M. Novick. Fast parallel algorithms for the modular decomposition. Technical report, Cornell University, 1989.

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

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

[69] R. Puzis, Y. Elovici, P. Zilberman, S. Dolev, and U. Brandes. Topology manipulations for speeding betweenness centrality computation. *J. of Complex Networks*, 3(1):84–112, 2014.

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

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

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

[73] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC'13*, pp 515–524.

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

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

[76] M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP'08*, pp 634–645.

[77] V. Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (Invited talk). In *IPEC'15*, pp 16–28.

[78] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS'10*, pp 645–654.

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