The Use of a Pruned Modular Decomposition for Maximum Matching Algorithms on Some Graph Classes

Guillaume Ducoffe
ICI – National Institute for Research and Development in Informatics, Bucharest, Romania
The Research Institute of the University of Bucharest ICUB, Bucharest, Romania
guillaume.ducoffe@ici.ro

Alexandru Popa
University of Bucharest, Bucharest, Romania
ICI – National Institute for Research and Development in Informatics, Bucharest, Romania
alexandru.popa@fmi.unibuc.ro

Abstract

We address the following general question: given a graph class $\mathcal{C}$ on which we can solve MAXIMUM MATCHING in (quasi) linear time, does the same hold true for the class of graphs that can be *modularly decomposed* into $\mathcal{C}$? As a way to answer this question for distance-hereditary graphs and some other superclasses of cographs, we study the combined effect of modular decomposition with a pruning process over the quotient subgraphs. We remove sequentially from all such subgraphs their so-called one-vertex extensions (i.e., pendant, anti-pendant, twin, universal and isolated vertices). Doing so, we obtain a "pruned modular decomposition", that can be computed in quasi linear time. Our main result is that if all the pruned quotient subgraphs have bounded order then a maximum matching can be computed in linear time. The latter result strictly extends a recent framework in (Coudert et al., SODA'18). Our work is the first to explain why the existence of some nice ordering over the *modules* of a graph, instead of just over its vertices, can help to speed up the computation of maximum matchings on some graph classes.

2012 ACM Subject Classification Mathematics of computing → Graph theory, Theory of computation → Design and analysis of algorithms

Keywords and phrases maximum matching, FPT in P, modular decomposition, pruned graphs, one-vertex extensions, P_4-structure

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.6

Related Version A full version of the paper is available at [14], `https://arxiv.org/abs/1804.09407`.

Funding This work was supported by the Institutional research programme PN 1819 "Advanced IT resources to support digital transformation processes in the economy and society - RESINFO-TD" (2018), project PN 1819-01-01 "Modeling, simulation, optimization of complex systems and decision support in new areas of IT&C research", funded by the Ministry of Research and Innovation, Romania.

1 Introduction

Can we compute a maximum matching in a graph in linear-time? – i.e., computing a maximum set of pairwise disjoint edges in a graph. – Despite considerable years of research and the design of elegant combinatorial and linear programming techniques, the best-known algorithms for this fundamental problem have stayed blocked to an $\mathcal{O}(m\sqrt{n})$-time complexity on n-vertex m-edge graphs [22]. Nevertheless, we can use some well-structured graph classes in order to overcome this superlinear barrier for particular cases of graphs. Our work combines two successful approaches for this problem, namely, the use of a *vertex-ordering* characterization for certain graph classes [5, 10, 21], and a recent technique based on the decomposition of a graph by its *modules* [9]. We detail these two approaches in what follows, before summarizing our contributions.

1.1 Related work

A cornerstone of most Maximum Matching algorithms is the notion of augmenting paths [2, 15]. However, although we can compute a set of augmenting paths in linear-time [16], this is a tedious task that involves the technical notion of blossoms and this may need to be repeated $\Omega(\sqrt{n})$ times before a maximum matching can be computed [19]. A well-known greedy approach consists in, given some total ordering $(v_1, v_2, \ldots, v_n)$ over the vertices in the graph, to consider the exposed vertices v_i by increasing order, then to try to match them with some exposed neighbour v_j that appears later in the ordering [12]. The vertex v_j can be chosen either arbitrarily or according to some specific rules depending on the graph class we consider. Our initial goal was to extend similar reduction rules to *module-orderings*.

Modular decomposition. A module in a graph $G = (V, E)$ is any vertex-subset X such that every vertex of $V \setminus X$ is either adjacent to every of X or nonadjacent to every of X. The *modular decomposition* of G is a recursive decomposition of G according to its modules [18]. We postpone its formal definition until Section 2. For now, we only want to stress that the vertices in the "quotient subgraphs" that are outputted by this decomposition represent modules of G (*e.g.*, see Fig. 1 for an insightful illustration). Our main motivation for considering modular decomposition in this note is its recent use in the field of parameterized complexity for *polynomial* problems. More precisely, let us call *modular-width* of a graph G the minimum $k \geq 2$ such that every quotient subgraph in the modular decomposition of G is either "degenerate" (i.e., complete or edgeless) or of order at most k. With Coudert, we proved in [9] that many "hard" graph problems in P – for which no linear-time algorithm is likely to exist – can be solved in $k^{\mathcal{O}(1)}(n + m)$-time on graphs with modular-width at most k. In particular, we proposed an $\mathcal{O}(k^4 n + m)$-time algorithm for Maximum Matching.

One appealing aspect of our approach in [9] was that, for most problems studied, we obtained a linear-time reduction from the input graph G to some (smaller) quotient subgraph G' in its modular decomposition. – We say that the problem is preserved by quotient. – This paved the way to the design of efficient algorithms for these problems on graph classes with *unbounded* modular-width, assuming their quotient subgraphs are simple enough w.r.t. the problem at hands. We illustrated this possibility through the case of $(q, q-3)$-graphs (i.e., graphs where no set of at most q vertices, $q \geq 7$, can induce more than $q - 3$ paths of length four). However, this approach completely fell down for Maximum Matching. Indeed, our Maximum Matching algorithm in [9] works on supergraphs of the quotient graphs that need to be repeatedly updated every time a new augmenting path is computed. Such approach did not help much in exploiting the structure of quotient graphs. We managed to do so for $(q, q-3)$-graphs only through the help of a deeper structural theorem on the nontrivial modules in this class of graphs. Nevertheless, to take a shameful example, it was not even known before this work whether Maximum Matching could be solved faster than with the state-of-the art algorithms on graphs that can be modularly decomposed into *paths*!

1.2 Our contributions

We propose *pruning rules* on the modules in a graph (some of them new and some others revisited) that can be used in order to compute Maximum Matching in linear-time on several new graph classes. More precisely, given a module M in a graph $G = (V, E)$, recall that M is corresponding to some vertex v_M in a quotient graph G' of the modular decomposition of G. Assuming v_M is a so-called *one-vertex extension* in G' (i.e., it is pendant, anti-pendant, universal, isolated or it has a twin), we show that a maximum matching for G can be computed from a maximum matching of $G[M]$ and a maximum matching of $G \setminus M$ efficiently (see Section 4). Our rules are purely *structural*, in the sense that they only rely on the structural properties of v_M in G' and not on any additional assumption on the nontrivial modules. Some of these rules (*e.g.*, for isolated or universal modules) were first introduced in [9] – although with slightly different correctness proofs. Our main technical contributions in this work are the pruning rules for, respectively, *pendant* and *anti-pendant* modules (see Sections 4.2 and 4.3). The latter two cases are surprisingly the most intricate. In particular, they require amongst other techniques: the computation of specified augmenting paths of length up to 7, the addition of some "virtual edges" in other modules, and a careful swapping between some matched and unmatched edges.

Then, we are left with pruning every quotient subgraph in the modular decomposition by sequentially removing the one-vertex extensions. We prove that the resulting "pruned quotient subgraphs" are unique (independent from the removal orderings) and that they can be computed in quasi linear-time using a *trie* data-structure (Section 3). Furthermore, as a case-study we prove that several superclasses of cographs are totally decomposable w.r.t. this new "pruned modular decomposition". These classes are further discussed in Section 5. Note that for some of them, such as distance-hereditary graphs, we so obtain the first known linear-time algorithm for Maximum Matching – thereby extending previous partial results obtained for bipartite and chordal distance-hereditary graphs [10]. Our approach actually has similarities with a general greedy scheme applied to distance-hereditary graphs [7]. With slightly more work, we can extend our approach to every graph that can be modularly decomposed into cycles. The case of graphs of bounded *modular treewidth* [23] is left as an interesting open question.

Definitions and our first results are presented in Section 2. We introduce the pruned modular decomposition in Section 3, where we show that it can be computed in quasi linear-time. Then, the core of the paper is Section 4 where the pruning rules are presented along with their correctness proofs. In particular, we state our main result in Section 4.4. Applications of our approach to some graph classes are discussed in Section 5. Finally, we conclude in Section 6 with some open questions. Due to lack of space, several proofs are omitted. Full proofs can be found in our technical report [14].

2 Preliminaries

For the standard graph terminology, see [3]. We only consider graphs that are finite, simple and unweighted. For any graph $G = (V, E)$ let $n = |V|$ and $m = |E|$. Given a vertex $v \in V$, we denote its (open) neighbourhood by $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ and its closed neighbourhood by $N_G[v] = N_G(v) \cup \{v\}$. Similarly, we define the neighbourhood of any vertex-subset $S \subseteq V$ as $N_G(S) = \left(\bigcup_{v \in S} N_G(v)\right) \setminus S$. In what follows, we introduce our main algorithmic tool for the paper as well as the graph problems we study.

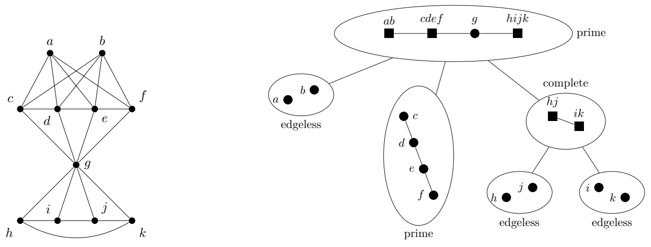

Figure 1 A graph and its modular decomposition.

Modular decomposition

A *module* in a graph $G = (V, E)$ is any subset $M \subseteq V(G)$ such that for any $u, v \in M$ we have $N_G(v) \setminus M = N_G(u) \setminus M$. There are trivial examples of modules such as $\emptyset$, V, and $\{v\}$ for every $v \in V$. Let $\mathcal{P} = \{M_1, M_2, \ldots, M_p\}$ be a partition of the vertex-set V. If for every $1 \leq i \leq p$, M_i is a module of G, then we call $\mathcal{P}$ a *modular partition* of G. By abuse of notation, we will sometimes identify a module M_i with the induced subgraph $H_i = G[M_i]$, i.e., we will write $\mathcal{P} = \{H_1, H_2, \ldots H_p\}$. The *quotient subgraph* $G/\mathcal{P}$ has vertex-set $\mathcal{P}$, and there is an edge between every two modules $M_i, M_j \in \mathcal{P}$ such that $M_i \times M_j \subseteq E$. Conversely, let $G' = (V', E')$ be a graph and let $\mathcal{P} = \{H_1, H_2, \ldots H_p\}$. be a collection of subgraphs. The *substitution graph* $G'(\mathcal{P})$ is obtained from G' by replacing every vertex $v_i \in V'$ with a module inducing H_i. In particular, for $G' =^{\text{def}} G/\mathcal{P}$ we have that $G'(\mathcal{P}) = G$.

We say that G is *prime* if its only modules are trivial (i.e., $\emptyset$, V, and the singletons $\{v\}$). We call a module M *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. Let $\mathcal{M}(G)$ be the family of all inclusion wise maximal strong modules of G that are proper subsets of V. The family $\mathcal{M}(G)$ is a modular partition of G [18], and so, we can define $G' = G/\mathcal{M}(G)$. The following structure theorem is due to Gallai.

▶ **Theorem 1** ([17]). *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 $G' = G/\mathcal{M}(G)$ is prime for modular decomposition.*

We now formally define the modular decomposition of G – introduced earlier in Section 1. We output the quotient graph $G' = G/\mathcal{M}(G)$ and, for any strong module $M \in \mathcal{M}(G)$ that is nontrivial (possibly none if $G = G'$), we also output the modular decomposition of $G[M]$. By Theorem 1 the subgraphs from the modular decomposition are either edgeless, complete, or prime for modular decomposition. See Fig. 1 for an example. The modular decomposition of a given graph $G = (V, E)$ can be computed in linear-time [25]. There are many graph classes that can be characterized using the modular decomposition.In particular, G is a cograph if and only if every quotient subgraph in its modular decomposition is either complete or disconnected [8].

Maximum Matching

A matching in a graph is defined as a set of edges with pairwise disjoint end vertices. The maximum cardinality of a matching in a given graph $G = (V, E)$ is denoted by $\mu(G)$.

▶ **Problem 2** (Maximum Matching).
Input: *A graph $G = (V, E)$.*
Output: *A matching of G with maximum cardinality.*

We remind the reader that Maximum Matching can be solved in $\mathcal{O}(m\sqrt{n})$-time on general graphs [22] – although we do not use this result directly in our paper. Furthermore, let $G = (V, E)$ be a graph and let $F \subseteq E$ be a matching of G. We call a vertex matched if it is incident to an edge of F, and exposed otherwise. Then, we define an F-augmenting path as a path where the two ends are exposed, and the edges belong alternatively to F and not to F. It is well-known and easy to check that, given an F-augmenting path P, the matching $E(P)\Delta F$ (obtained by symmetric difference on the edges) has larger cardinality than F.

▶ **Lemma 3** (Berge, [2]). *A matching F in $G = (V, E)$ is maximum if and only if there is no F-augmenting path.*

In this paper, we will consider an intermediate matching problem, first introduced in [9].

▶ **Problem 4** (Module Matching).
Input: *A graph $G' = (V', E')$ with the following additional information;*

- *a collection of subgraphs $\mathcal{P} = \{H_1, H_2, \ldots, H_p\}$;*
- *a collection $\mathcal{F} = \{F_1, F_2, \ldots, F_p\}$,*
  with F_i being a maximum matching of H_i for every i.

Output: *A matching of $G = G'(\mathcal{P})$ with maximum cardinality.*

A natural choice for Module Matching would be to take $\mathcal{P} = \mathcal{M}(G)$. However, we will allow $\mathcal{P}$ to take different values for our reduction rules.

Additional notations. Let $\langle G', \mathcal{P}, \mathcal{F}\rangle$ be any instance of Module Matching. The order of G', equivalently the cardinality of $\mathcal{P}$, is denoted by p. For every $1 \le i \le p$ let $M_i = V(H_i)$ and let $n_i = |M_i|$ be the order of H_i. We denote $\delta_i = |E(M_i, \overline{M_i})|$ the size of the cut $E(M_i, \overline{M_i})$ with all the edges between M_i and $N_G(M_i)$. In particular, we have $\delta_i = \sum_{v_j \in N_{G'}(v_i)} n_i n_j$. Let us define $\Delta m(G') = \sum_{i=1}^{p} \delta_i$. We will omit the dependency in G' if it is clear from the context. Finally, let $\Delta\mu = \mu(G) - \sum_{i=1}^{p} \mu(H_i)$.

Our framework is based on the following lemma (inspired from [9]).

▶ **Lemma 5.** *Let $G = (V, E)$ be a graph. Suppose that for every H' in the modular decomposition of G we can solve* Module Matching *on any instance $\langle H', \mathcal{P}, \mathcal{F}\rangle$ in time $T(p, \Delta m, \Delta\mu)$, where T is a subadditive function*[1]*. Then, we can solve* Maximum Matching *on G in time $\mathcal{O}(T(\mathcal{O}(n), m, n))$.*

An important observation for our subsequent analysis is that, given any module M of a graph G, the internal structure of $G[M]$ has no more relevance after we computed a maximum matching F_M for this subgraph. More precisely, we will use the following lemma:

▶ **Lemma 6** ([9]). *Let M be a module of $G = (V, E)$, let $G[M] = (M, E_M)$ and let $F_M \subseteq E_M$ be a maximum matching of $G[M]$. Then, every maximum matching of $G'_M = (V, (E \setminus E_M) \cup F_M)$ is a maximum matching of G.*

By Lemma 6 we can modify our algorithmic framework as follows. For every instance $\langle G', \mathcal{P}, \mathcal{F}\rangle$ for Module Matching, we can assume that $H_i = (M_i, F_i)$ for every $1 \le i \le p$.

[1] We stress that every polynomial function is subadditive.

Data structures. Finally, let $\langle G', \mathcal{P}, \mathcal{F} \rangle$ be any instance for Module Matching. A *canonical ordering* of H_i (w.r.t. F_i) is a total ordering over $V(H_i)$ such that the exposed vertices appear first, and every two vertices that are matched together are consecutive. In what follows, we will assume that we have access to a canonical ordering for every i. Such orderings can be computed in time $\mathcal{O}(\sum_i |M_i| + |F_i|)$ by scanning all the modules and the matchings in $\mathcal{F}$, that is an $\mathcal{O}(\Delta m)$ provided G' has no isolated vertex.

Furthermore, let F be a (not necessarily maximum) matching for the subdivision $G = G'(\mathcal{P})$. We will make the standard assumption that, for every $v \in V(G)$, we can decide in constant-time whether v is matched by F, and if so, we can also access in constant-time to the vertex matched with v.

3 A pruned modular decomposition

In this section, we introduce a pruning process over the quotient subgraphs, that we use in order to refine the modular decomposition.

▶ **Definition 7.** Let $G = (V, E)$ be a graph. We call $v \in V$ a one-vertex extension if it falls in one of the following cases:
- $N_G[v] = V$ (*universal*) or $N_G(v) = \emptyset$ (*isolated*);
- $N_G[v] = V \setminus u$ (*anti-pendant*) or $N_G(v) = \{u\}$ (*pendant*), for some $u \in V \setminus v$;
- $N_G[v] = N_G[u]$ (*true twin*) or $N_G(v) = N_G(u)$ (*false twin*), for some $u \in V \setminus v$.

A pruned subgraph of G is obtained from G by sequentially removing one-vertex extensions (in the current subgraph) until it can no more be done. This terminology was introduced in [20], where they only considered the removals of twin and pendant vertices. Also, the clique-width of graphs that are totally decomposed by the above pruning process (i.e., with their pruned subgraph being a singleton) was studied in [24] [2]. Our contribution in this part is twofold. First, we show that the gotten subgraph is "almost" independent of the removal ordering, i.e., there is a unique pruned subgraph of G (up to isomorphism). The latter can be derived from the following (easy) lemma:

▶ **Lemma 8.** *Let $G = (V, E)$ be a graph and let $v, v' \in V$ be one-vertex extensions of G. If v, v' are* not *pairwise twins then v' is a one-vertex extension of $G \setminus v$.*

▶ **Corollary 9.** *Every graph $G = (V, E)$ has a unique pruned subgraph up to isomorphism.*

For many graph classes a pruning sequence can be computed in linear-time. We observe that the same can be done for any graph (up to a logarithmic factor).

▶ **Proposition 10.** *For every graph $G = (V, E)$, we can compute a pruned subgraph in $\mathcal{O}(n + m \log n)$-time.*

Proof. By Corollary 9, we are left with greedily searching for, then eliminating, the one-vertex extensions. We can compute the ordered degree sequence of G in $\mathcal{O}(n+m)$-time. Furthermore, after any vertex v is eliminated, we can update this sequence in $\mathcal{O}(|N(v)|)$-time. Hence, up to a total update time in $\mathcal{O}(n + m)$, at any step we can detect and remove in constant-time any vertex that is either universal, isolated, pendant or anti-pendant. Finally, in [20] they proposed a trie data-structure supporting the following two operations: suppression of a vertex; and detection of true or false twins (if any). The total time for all the operations on this data-structure is in $\mathcal{O}(n + m \log n)$ [20]. ◀

[2] Anti-twins are also defined as one-vertex extensions in [24]. Their integration to this framework remains to be done.

We will term "pruned modular decomposition" of a graph G the collection of the pruned subgraphs for all the quotient subgraphs in the modular decomposition of G. Note that there is a unique pruned modular decomposition of G (up to isomorphism) and that it can be computed in $\mathcal{O}(n + m \log n)$-time by Proposition 10 (applied to every quotient subgraph in the modular decomposition separately). Furthermore, we remark that most cases of one-vertex extensions imply the existence of non trivial modules, and so, they cannot exist in the prime quotient subgraphs of the modular decomposition. Nevertheless, such vertices may appear after removal of pendant or anti-pendant vertices (*e.g.*, in the bull graph).

4 Reduction rules

Let $\langle G', \mathcal{P}, \mathcal{F} \rangle$ be any instance of Module Matching. Suppose that v_1, the vertex corresponding to M_1 in G', is a one-vertex extension. Under this assumption, we present reduction rules to a smaller instance $\langle G^*, \mathcal{P}^*, \mathcal{F}^* \rangle$ where $|\mathcal{P}^*| < |\mathcal{P}|$. Each rule can be implemented to run in $\mathcal{O}(\Delta m(G') - \Delta m(G^*))$-time. Due to lack of space, we skip the complexity analysis.

In Section 4.1 we recall the rules introduced in [9] for universal and isolated modules (explicitly) and for false or true twin modules (implicitly). Our main technical contributions are the reduction rules for pendant and anti-pendant modules (in Sections 4.2 and 4.3, respectively), which are surprisingly the most intricate. Finally, we end this section stating our main result (Theorem 29).

4.1 Simple cases

We introduce two local operations on a matching, first used in [26] for cographs. Let $F \subseteq E$ be a matching and let $M \subseteq V$ be a module.

▶ **Operation 11** (MATCH). *While there are $x \in M,\ y \in N(M)$ exposed, add $\{x, y\}$ to F.*

▶ **Operation 12** (SPLIT). *While there are $x, x' \in M,\ y, y' \in N(M)$ such that x and x' are exposed, and $\{y, y'\} \in F$, replace $\{y, y'\}$ in F by $\{x, y\},\ \{x', y'\}$.*

Let $G = H_1 \oplus H_2$ be the join of the two graphs H_1, H_2 and let F_1, F_2 be maximum matchings for H_1, H_2, respectively. The "MATCH and SPLIT" technique consists in applying Operations 11 then 12 to $M = V(H_1)$ and $F = F_1 \cup F_2$, thereby obtaining a new matching F', then to $M = V(H_2)$ and $F = F'$. Based on this technique, we design the following rules:

▶ **Reduction rule 13** (see also [9]). *Suppose v_1 is isolated in G'. We set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$, and $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Furthermore, let F^* be a maximum matching of $G^*(\mathcal{P}^*) = G[V \setminus M_1]$. We output $F^* \cup F_1$.*

▶ **Reduction rule 14** (see also [9]). *Suppose v_1 is universal in G'. We set $G^* = G \setminus v_1$, $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$, $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Furthermore, let F^* be a maximum matching of the subdivision $G^*(\mathcal{P}^*) = G[V \setminus M_1]$. We apply the "*MATCH *and* SPLIT*" technique to M_1, F_1 with $V \setminus M_1, F^*$.*

▶ **Reduction rule 15.** *Suppose v_1, v_2 are false twins in G'. We set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \{H_1 \cup H_2\} \cup (\mathcal{P} \setminus \{H_1, H_2\})$, $\mathcal{F}^* = \{F_1 \cup F_2\} \cup (\mathcal{F} \setminus \{F_1, F_2\})$. We output a maximum matching of $G^*(\mathcal{P}^*) = G$.*

▶ **Reduction rule 16.** *Suppose v_1, v_2 are true twins in G'. Let F_2^* be the matching of $H_1 \oplus H_2$ obtained from the "*MATCH *and* SPLIT*" technique applied to M_1, F_1 with M_2, F_2. We set $G^* = G \setminus v_1$, $\mathcal{P}^* = \{H_1 \oplus H_2\} \cup (\mathcal{P} \setminus \{H_1, H_2\})$, $\mathcal{F}^* = \{F_2^*\} \cup (\mathcal{F} \setminus \{F_1, F_2\})$. We output a maximum matching of $G^*(\mathcal{P}^*) = G$.*

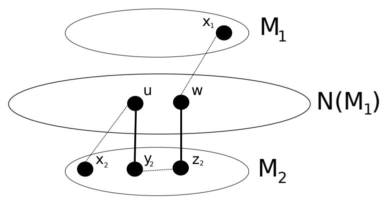

Figure 2 An augmenting path of length 5 with ends x_1, x_2. Matched edges are drawn in bold.

4.2 Anti-pendant

Suppose v_1 is anti-pendant in G'. W.l.o.g., v_2 is the unique vertex that is nonadjacent to v_1 in G'. By Lemma 6, we can also assume w.l.o.g. that $E(H_i) = F_i$ for every i. In this situation, we start applying Reduction rule 13, i.e., we set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$, $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Then, we obtain a maximum matching F^* of $G \setminus M_1$ (i.e., by applying our reduction rules to this new instance). Finally, from F_1 and F^*, we compute a maximum matching F of G, using an intricate procedure. We detail this procedure next.

First phase: pre-processing. Our correctness proofs in what follows will assume that some additional properties hold on the matched vertices in F^*. So, we start correcting the initial matching F^* so that it is the case. For that, we introduce two "swapping" operations. Recall that v_2 is the unique vertex that is nonadjacent to v_1 in G'.

▶ **Operation 17** (REPAIR). *While there exist $x_2, y_2 \in M_2$ such that $\{x_2, y_2\} \in F_2$ and y_2 is exposed in F^*, we replace any edge $\{x_2, w\} \in F^*$ by $\{x_2, y_2\}$.*

▶ **Operation 18** (ATTRACT). *While there exist $x_2 \in M_2$ exposed and $\{u, w\} \in F^*$ such that $u \in N_G(M_2), w \notin M_2$, we replace $\{u, w\}$ by $\{u, x_2\}$.*

Let $F^{(0)} = F_1 \cup F^*$. Summarizing, we get:

▶ **Definition 19.** A matching F of G is *good* if it satisfies the following two properties:
1. every vertex matched by $F_1 \cup F_2$ is also matched by F;
2. either every vertex in M_2 is matched, or there is no matched edge in $N_G(M_2) \times N_G(M_1)$.

▶ **Fact 20.** *$F^{(0)}$ is a good matching of G.*

Main phase: a modified Match and Split. We now apply the following three operations sequentially:
1. Match$(M_1, F^{(0)})$ (Operation 11). Doing so, we obtain a larger good matching $F^{(1)}$.
2. Split$(M_1, F^{(1)})$ (Operation 12). Doing so, we obtain a larger good matching $F^{(2)}$.
3. the operation Unbreak, defined in what follows (see also Fig. 2 for an illustration):
   ▶ **Operation 21** (Unbreak). *While there exist $x_1 \in M_1$ and $x_2 \in M_1 \cup M_2$ exposed, and there also exist $\{y_2, z_2\} \in F_2 \setminus F^{(2)}$, we replace any two edges $\{y_2, u\}, \{z_2, w\} \in F^{(2)}$ by the three edges $\{x_2, u\}, \{y_2, z_2\}$ and $\{w, x_1\}$.*
   We stress that the two edges $\{y_2, u\}, \{z_2, w\} \in F^{(2)}$ always exist since $F^{(2)}$ is a good matching of G. Furthermore doing so, we obtain a larger matching $F^{(3)}$.

The resulting matching $F^{(3)}$ is not necessarily maximum. However, this matching satisfies the following crucial property:

▶ **Lemma 22.** *No vertex of M_1 can be an end in an $F^{(3)}$-augmenting path.*

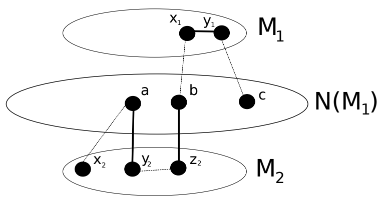

Figure 3 An augmenting path of length 7 with ends x_2, c. Matched edges are drawn in bold.

Finalization phase: breaking some edges in F_1. Intuitively, the matching $F^{(3)}$ may not be maximum because we sometimes need to borrow some edges of F_1 in augmenting paths. So, we complete our procedure by performing the following two operations: Let U_1 contain all the exposed vertices in $N(M_1)$. Consider the subgraph $G[M_1 \cup U_1] = G[M_1] \oplus G[U_1]$. The set U_1 is a module of this subgraph. We apply Split$(U_1, F^{(3)})$ in $G[M_1 \cup U_1]$. Doing so, we obtain a larger good matching $F^{(4)}$. Then, we apply LocalAug, defined next (see also Fig. 3 for an illustration):

▶ **Operation 23** (LocalAug). *While there exist $x_2 \in M_2$ and $c \in N(M_1)$ exposed, and there also exist $\{x_1, y_1\} \in F_1 \cap F^{(4)}$ and $\{y_2, z_2\} \in F_2 \setminus F^{(4)}$, we do the following:*
- *we remove $\{x_1, y_1\}$ and any edge $\{a, y_2\}, \{b, z_2\}$ from $F^{(4)}$;*
- *we add $\{x_2, a\}, \{y_2, z_2\}, \{b, x_1\}$ and $\{y_1, c\}$ in $F^{(4)}$.*

We stress that the two edges $\{y_2, a\}, \{z_2, b\} \in F^{(4)}$ always exist since $F^{(4)}$ is a good matching of G. Furthermore doing so, we obtain a larger matching $F^{(5)}$.

▶ **Lemma 24.** *$F^{(5)}$ is a maximum-cardinality matching of G.*

4.3 Pendant

Suppose v_1 is pendant in G'. W.l.o.g., v_2 is the unique vertex that is adjacent to v_1 in G'. This last case is arguably more complex than the others since it requires both a pre-processing and a post-processing treatment on the matching.

First phase: greedy matching. We apply the "Match and Split" technique to M_1. Doing so, we obtain a set $F_{1,2}$ of matched edges between M_1 and M_2. We remove $V(F_{1,2})$, the set of vertices incident to an edge of $F_{1,2}$, from G. Then, there are three cases. If $M_2 \subseteq V(F_{1,2})$ then $M_1 \setminus V(F_{1,2})$ is isolated. We apply Reduction rule 13. If $M_1 \subseteq V(F_{1,2})$ then M_1 is already eliminated. The interesting case is when both $M_1 \setminus V(F_{1,2})$ and $M_2 \setminus V(F_{1,2})$ are nonempty. In particular, suppose there remains an exposed vertex $x_1 \in M_1 \setminus V(F_{1,2})$. Since $M_2 \setminus V(F_{1,2}) \neq \emptyset$, there exists $\{x_2, y_2\} \in F_2$ such that $x_2, y_2 \notin V(F_{1,2})$. We remove x_1 from M_1, x_2 from M_2, $\{x_2, y_2\}$ from F_2 and then we add $\{x_1, x_2\}$ in $F_{1,2}$. Our first result in this section is that there always exists an optimal solution that contains $F_{1,2}$. This justifies a posteriori the removal of $V(F_{1,2})$ from G.

▶ **Lemma 25.** *There is a maximum matching of G that contains all edges in $F_{1,2}$.*

We stress that during this phase, all the operations except maybe the last one increase the cardinality of the matching. Furthermore, the only possible operation that does not increase the cardinality of the matching is the replacement of an edge in F_2 by an edge in $F_{1,2}$. Doing so, either we fall in one of the two pathological cases $M_1 \subseteq V(F_{1,2})$ or $M_2 \subseteq V(F_{1,2})$ (easy to solve), or then we obtain through the replacement operation the following stronger property:

▶ **Property 26.** *All vertices in M_1 are matched by F_1.*

We will assume Property 26 to be true for the remaining of this section.

Second phase: virtual split edges. We complete the previous phase by performing a SPLIT between M_2, M_1 (Operation 12). That is, while there exist two exposed vertices $x_2, y_2 \in M_2$ and a matched edge $\{x_1, y_1\} \in F_1$ we replace $\{x_1, y_1\}$ by $\{x_1, x_2\}, \{y_1, y_2\}$ in the current matching. However, we encode the SPLIT operation using virtual edges in H_2. Formally, we add a virtual edge $\{x_2, y_2\}$ in H_2 that is labeled by the corresponding edge $\{x_1, y_1\} \in F_1$. Let H_2^* and F_2^* be obtained from H_2 and F_2 by adding all the virtual edges. We set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \{H_2^*\} \cup (\mathcal{P} \setminus \{H_1, H_2\})$ and $\mathcal{F}^* = \{F_2^*\} \cup (\mathcal{F} \setminus \{F_1, F_2\})$.

Intuitively, virtual edges are used in order to shorten the augmenting paths crossing M_1.

Third phase: post-processing. Let F^* be a maximum-cardinality matching of the subdivision $G^*(\mathcal{P}^*)$ (i.e., obtained by applying our reduction rules to the new instance). We construct a matching F for G as follows. We add in F all the non virtual edges in F^*. For every virtual edge $\{x_2, y_2\}$, let $\{x_1, y_1\} \in F_1$ be its label. If $\{x_2, y_2\} \in F^*$ then we add $\{x_1, y_2\}, \{x_2, y_1\}$ in F, otherwise we add $\{x_1, y_1\}$ in F. In the first case, we say that we confirm the SPLIT operation, whereas in the second case we say that we cancel it. Finally, we complete F with all the edges of F_1 that do not label any virtual edge (i.e., unused during the second phase).

▶ **Lemma 27.** *F is a maximum-cardinality matching of G.*

The above result is proved by contrapositive. More precisely, we prove intricate properties on the intersection of *shortest* augmenting paths with pendant modules. Using these properties and the virtual edges, we could transform any shortest F-augmenting path into an F^*-augmenting path, a contradiction.

4.4 Main result

Our framework consists in applying any reduction rule presented in this section until it can no more be done. Then, we rely on the following result:

▶ **Theorem 28** ([9]). *We can solve* MODULE MATCHING *for $\langle G', \mathcal{P}, \mathcal{F} \rangle$ in $\mathcal{O}(\Delta\mu \cdot p^4)$-time.*

We are now ready to state our main result in this paper (the proof of which directly follows from all the previous results in this section).

▶ **Theorem 29.** *Let $G = (V, E)$ be a graph. Suppose that, for every prime subgraph H' in the modular decomposition of G, its pruned subgraph has order at most k. Then, we can solve* MAXIMUM MATCHING *for G in $\mathcal{O}(k^4 \cdot n + m \log n)$-time.*

5 Applications

We conclude this paper presenting applications and refinements of our main result to some graph classes. Recall that cographs are exactly the graphs that are totally decomposable by modular decomposition [8]. We start showing that several distinct generalizations of cographs in the literature are totally decomposable by the pruned modular decomposition.

Distance-hereditary graphs. A graph $G = (V, E)$ is distance-hereditary if it can be reduced to a singleton by pruning sequentially the pendant vertices and twin vertices [1]. Conversely, G is co-distance hereditary if it is the complement of a distance-hereditary graph, i.e., it can be reduced to a singleton by pruning sequentially the anti-pendant vertices and twin vertices. In both cases, the corresponding pruning sequence can be computed in linear-time [11, 13]. Therefore, we can derive the following result from our framework:

▶ **Proposition 30.** *We can solve* Maximum Matching *in linear-time on graphs that can be modularly decomposed into distance-hereditary graphs and co-distance hereditary graphs.*

Trees are a special subclass of distance-hereditary graphs. We say that a graph has *modular treewidth* at most k if every prime quotient subgraph in its modular decomposition has treewidth at most k. In particular, graphs with modular treewidth at most one are exactly the graphs that can be modularly decomposed into trees[3]. We stress the following consequence of Proposition 30:

▶ **Corollary 31.** *We can solve* Maximum Matching *in linear-time on graphs with modular-treewidth at most one.*

The case of graphs with modular treewidth $k \geq 2$ is left as an intriguing open question.

Tree-perfect graphs. Two graphs G_1, G_2 are P_4-isomorphic if there exists a bijection from G_1 to G_2 such that a 4-tuple induces a P_4 in G_1 if and only if its image in G_2 also induces a P_4 [6]. The notion of P_4-isomorphism plays an important role in the study of perfect graphs. A graph is *tree-perfect* if it is P_4-isomorphic to a tree [4]. We prove the following result:

▶ **Proposition 32.** *Tree-perfect graphs are totally decomposable by the pruned modular decomposition. In particular, we can solve* Maximum Matching *in linear-time on tree-perfect graphs.*

Our proof is based on a deep structural characterization of tree-perfect graphs [4].

The case of unicycles. We end up this section with a refinement of our framework for the special case of unicyclic quotient graphs (*a.k.a.*, graphs with exactly one cycle).

▶ **Proposition 33.** *We can solve* Maximum Matching *in linear-time on the graphs that can be modularly decomposed into unicycles.*

For that, we reduce the case of unicycles to the case of cycles (removing pendant modules). Then, we test for all possible numbers of matched edges between two adjacent modules. Doing so, we reduce the case of cycles to the case of paths.

6 Open problems

The pruned modular decomposition happens to be an interesting add up in the study of Maximum Matching algorithms. An exhaustive study of its other algorithmic applications remains to be done. Moreover, another interesting question is to characterize the graphs that are totally decomposable by this new decomposition. We note that our pruning process can

[3] Our definition is more restricted than the one in [23] since they only impose the quotient subgraph G' to have bounded treewidth.

be seen as a repeated update of the modular decomposition of a graph after some specified modules (pendant, anti-pendant) are removed. However, we can only detect a restricted family of these new modules (i.e., universal, isolated, twins). A fully dynamic modular decomposition algorithm could be helpful in order to further refine our framework.

References

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

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

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

4 A. Brandstädt and V. Le. Tree-and forest-perfect graphs. *Discrete applied mathematics*, 95(1-3):141–162, 1999.

5 M. Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *ISAAC*, pages 146–155. Springer, 1996.

6 V. Chvátal. A semi-strong perfect graph conjecture. In *North-Holland mathematics studies*, volume 88, pages 279–280. Elsevier, 1984.

7 O. Cogis and E. Thierry. Computing maximum stable sets for distance-hereditary graphs. *Discrete Optimization*, 2(2):185–188, 2005.

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

9 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *SODA'18*, pages 2765–2784. SIAM, 2018.

10 E. Dahlhaus and M. Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84(1-3):79–91, 1998.

11 G. Damiand, M. Habib, and C. Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–111, 2001.

12 F. Dragan. On greedy matching ordering and greedy matchable graphs. In *WG'97*, volume 1335 of *LNCS*, pages 184–198. Springer, 1997.

13 S. Dubois, V. Giakoumakis, and C. El Mounir. On co-distance hereditary graphs. In *CTW*, pages 94–97, 2008.

14 G. Ducoffe and A. Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. Technical Report arXiv:1804.09407, arXiv, 2018.

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

16 H. Gabow and R. Tarjan. A linear-time algorithm for a special case of disjoint set union. In *STOC'83*, pages 246–251. ACM, 1983.

17 Tibor Gallai. Transitiv orientierbare graphen. *Acta Math. Hungarica*, 18(1):25–66, 1967.

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

19 J. Hopcroft and R. Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

20 J. Lanlignel, O. Raynaud, and E. Thierry. Pruning graphs with digital search trees. application to distance hereditary graphs. In *STACS*, pages 529–541. Springer, 2000.

21 G. Mertzios, A. Nichterlein, and R. Niedermeier. Linear-time algorithm for maximum-cardinality matching on cocomparability graphs. *arXiv*, 2017. arXiv:1703.05598.

22 S. Micali and V. Vazirani. An $O(\sqrt{V}E)$ Algorithm for Finding Maximum Matching in General Graphs. In *FOCS'80*, pages 17–27. IEEE, 1980.

23 D. Paulusma, F. Slivovsky, and S. Szeider. Model counting for cnf formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016.

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

25 M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler Linear-Time Modular Decomposition Via Recursive Factorizing Permutations. In *ICALP*, pages 634–645. Springer, 2008.

26 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.