



HAL
open science

The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes

Guillaume Ducoffe, Alexandru Popa

► **To cite this version:**

Guillaume Ducoffe, Alexandru Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. 29th International Symposium on Algorithms and Computation (ISAAC 2018), Dec 2018, Jiaoxi, Yilan County, Taiwan. 10.4230/LIPIcs.ISAAC.2018.144 . hal-01955985

HAL Id: hal-01955985

<https://hal.science/hal-01955985>

Submitted on 14 Dec 2018

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

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

1 The use of a pruned modular decomposition for 2 MAXIMUM MATCHING algorithms on some graph 3 classes

4 **Guillaume Ducoffe**

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

6 The Research Institute of the University of Bucharest ICUB, Bucharest, Romania

7 guillaume.ducoffe@ici.ro

8 **Alexandru Popa**

9 University of Bucharest, Bucharest, Romania

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

11 alexandru.popa@fmi.unibuc.ro

12 — Abstract —

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

25 **2012 ACM Subject Classification** Graph theory, Design and analysis of algorithms

26 **Keywords and phrases** maximum matching; FPT in P; modular decomposition; pruned graphs;
27 one-vertex extensions; P_4 -structure

28 **Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2018.144

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

34 **1** Introduction

35 *Can we compute a maximum matching in a graph in linear-time?* – *i.e.*, computing a
36 maximum set of pairwise disjoint edges in a graph. – Despite considerable years of research
37 and the design of elegant combinatorial and linear programming techniques, the best-known
38 algorithms for this fundamental problem have stayed blocked to an $\mathcal{O}(m\sqrt{n})$ -time complexity
39 on n -vertex m -edge graphs [22]. Nevertheless, we can use some well-structured graph classes
40 in order to overcome this superlinear barrier for particular cases of graphs. Our work
41 combines two successful approaches for this problem, namely, the use of a *vertex-ordering*



© Guillaume Ducoffe and Alexandru Popa;
licensed under Creative Commons License CC-BY

29th International Symposium on Algorithms and Computation (ISAAC 2018).

Editors: Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao; Article No. 144; pp. 144:1–144:12

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

42 characterization for certain graph classes [5, 10, 21], and a recent technique based on the
 43 decomposition of a graph by its *modules* [9]. We detail these two approaches in what follows,
 44 before summarizing our contributions.

45 1.1 Related work

46 A cornerstone of most MAXIMUM MATCHING algorithms is the notion of augmenting paths [2,
 47 15]. However, although we can compute a set of augmenting paths in linear-time [16], this
 48 is a tedious task that involves the technical notion of blossoms and this may need to be
 49 repeated $\Omega(\sqrt{n})$ times before a maximum matching can be computed [19]. A well-known
 50 greedy approach consists in, given some total ordering (v_1, v_2, \dots, v_n) over the vertices in
 51 the graph, to consider the exposed vertices v_i by increasing order, then to try to match them
 52 with some exposed neighbour v_j that appears later in the ordering [12]. The vertex v_j can
 53 be chosen either arbitrarily or according to some specific rules depending on the graph class
 54 we consider. Our initial goal was to extend similar reduction rules to *module-orderings*.

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

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

83 1.2 Our contributions

84 We propose *pruning rules* on the modules in a graph (some of them new and some others
 85 revisited) that can be used in order to compute MAXIMUM MATCHING in linear-time on
 86 several new graph classes. More precisely, given a module M in a graph $G = (V, E)$,

87 recall that M is corresponding to some vertex v_M in a quotient graph G' of the modular
 88 decomposition of G . Assuming v_M is a so-called *one-vertex extension* in G' (i.e., it is
 89 pendant, anti-pendant, universal, isolated or it has a twin), we show that a maximum
 90 matching for G can be computed from a maximum matching of $G[M]$ and a maximum
 91 matching of $G \setminus M$ efficiently (see Section 4). Our rules are purely *structural*, in the sense
 92 that they only rely on the structural properties of v_M in G' and not on any additional
 93 assumption on the nontrivial modules. Some of these rules (e.g., for isolated or universal
 94 modules) were first introduced in [9] — although with slightly different correctness proofs.
 95 Our main technical contributions in this work are the pruning rules for, respectively, *pendant*
 96 and *anti-pendant* modules (see Sections 4.2 and 4.3). The latter two cases are surprisingly
 97 the most intricate. In particular, they require amongst other techniques: the computation
 98 of specified augmenting paths of length up to 7, the addition of some “virtual edges” in
 99 other modules, and a careful swapping between some matched and unmatched edges.

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

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

120 2 Preliminaries

121 For the standard graph terminology, see [3]. We only consider graphs that are finite, simple
 122 and unweighted. For any graph $G = (V, E)$ let $n = |V|$ and $m = |E|$. Given a vertex
 123 $v \in V$, we denote its (open) neighbourhood by $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ and its closed
 124 neighbourhood by $N_G[v] = N_G(v) \cup \{v\}$. Similarly, we define the neighbourhood of any
 125 vertex-subset $S \subseteq V$ as $N_G(S) = (\bigcup_{v \in S} N_G(v)) \setminus S$. In what follows, we introduce our main
 126 algorithmic tool for the paper as well as the graph problems we study.

127 Modular decomposition

128 A *module* in a graph $G = (V, E)$ is any subset $M \subseteq V(G)$ such that for any $u, v \in M$ we
 129 have $N_G(v) \setminus M = N_G(u) \setminus M$. There are trivial examples of modules such as \emptyset , V , and $\{v\}$
 130 for every $v \in V$. Let $\mathcal{P} = \{M_1, M_2, \dots, M_p\}$ be a partition of the vertex-set V . If for every
 131 $1 \leq i \leq p$, M_i is a module of G , then we call \mathcal{P} a *modular partition* of G . By abuse of

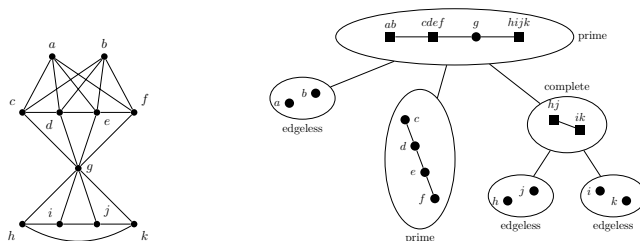
132 notation, we will sometimes identify a module M_i with the induced subgraph $H_i = G[M_i]$,
 133 *i.e.*, we will write $\mathcal{P} = \{H_1, H_2, \dots, H_p\}$. The *quotient subgraph* G/\mathcal{P} has vertex-set \mathcal{P} , and
 134 there is an edge between every two modules $M_i, M_j \in \mathcal{P}$ such that $M_i \times M_j \subseteq E$. Conversely,
 135 let $G' = (V', E')$ be a graph and let $\mathcal{P} = \{H_1, H_2, \dots, H_p\}$. be a collection of subgraphs.
 136 The *substitution graph* $G'(\mathcal{P})$ is obtained from G' by replacing every vertex $v_i \in V'$ with a
 137 module inducing H_i . In particular, for $G' =^{\text{def}} G/\mathcal{P}$ we have that $G'(\mathcal{P}) = G$.

138 We say that G is *prime* if its only modules are trivial (*i.e.*, \emptyset , V , and the singletons $\{v\}$).
 139 We call a module M *strong* if it does not overlap any other module, *i.e.*, for any module
 140 M' of G , either one of M or M' is contained in the other or M and M' do not intersect.
 141 Let $\mathcal{M}(G)$ be the family of all inclusion wise maximal strong modules of G that are proper
 142 subsets of V . The family $\mathcal{M}(G)$ is a modular partition of G [18], and so, we can define
 143 $G' = G/\mathcal{M}(G)$. The following structure theorem is due to Gallai.

144 ► **Theorem 1** ([17]). *For an arbitrary graph G exactly one of the following conditions is*
 145 *satisfied.*

- 146 1. G is disconnected;
- 147 2. its complement \overline{G} is disconnected;
- 148 3. or its quotient graph $G' = G/\mathcal{M}(G)$ is prime for modular decomposition.

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



■ **Figure 1** A graph and its modular decomposition.

158 Maximum Matching

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

► **Problem 1** (MAXIMUM MATCHING).

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

Output: A matching of G with maximum cardinality.

162 We remind the reader that MAXIMUM MATCHING can be solved in $\mathcal{O}(m\sqrt{n})$ -time on
 163 general graphs [22] — although we do not use this result directly in our paper. Furthermore,

164 let $G = (V, E)$ be a graph and let $F \subseteq E$ be a matching of G . We call a vertex matched if it
 165 is incident to an edge of F , and exposed otherwise. Then, we define an F -augmenting path
 166 as a path where the two ends are exposed, and the edges belong alternatively to F and not
 167 to F . It is well-known and easy to check that, given an F -augmenting path P , the matching
 168 $E(P)\Delta F$ (obtained by symmetric difference on the edges) has larger cardinality than F .

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

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

► **Problem 2** (MODULE MATCHING).

Input: A graph $G' = (V', E')$ with the following additional information;

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

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

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

175 **Additional notations.** Let $\langle G', \mathcal{P}, \mathcal{F} \rangle$ be any instance of MODULE MATCHING. The
 176 order of G' , equivalently the cardinality of \mathcal{P} , is denoted by p . For every $1 \leq i \leq p$ let
 177 $M_i = V(H_i)$ and let $n_i = |M_i|$ be the order of H_i . We denote $\delta_i = |E(M_i, \overline{M}_i)|$ the size
 178 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
 179 $\delta_i = \sum_{v_j \in N_{G'}(v_i)} n_i n_j$. Let us define $\Delta m(G') = \sum_{i=1}^p \delta_i$. We will omit the dependency in
 180 G' if it is clear from the context. Finally, let $\Delta \mu = \mu(G) - \sum_{i=1}^p \mu(H_i)$.

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

182 ► **Lemma 3.** *Let $G = (V, E)$ be a graph. Suppose that for every H' in the modular de-*
 183 *composition of G we can solve MODULE MATCHING on any instance $\langle H', \mathcal{P}, \mathcal{F} \rangle$ in time*
 184 *$T(p, \Delta m, \Delta \mu)$, where T is a subadditive function¹. Then, we can solve MAXIMUM MATCH-*
 185 *ING on G in time $\mathcal{O}(T(\mathcal{O}(n), m, n))$.*

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

189 ► **Lemma 4** ([9]). *Let M be a module of $G = (V, E)$, let $G[M] = (M, E_M)$ and let*
 190 *$F_M \subseteq E_M$ be a maximum matching of $G[M]$. Then, every maximum matching of $G'_M =$*
 191 *$(V, (E \setminus E_M) \cup F_M)$ is a maximum matching of G .*

192 By Lemma 4 we can modify our algorithmic framework as follows. For every instance
 193 $\langle G', \mathcal{P}, \mathcal{F} \rangle$ for MODULE MATCHING, we can assume that $H_i = (M_i, F_i)$ for every $1 \leq i \leq p$.

194 **Data structures.** Finally, let $\langle G', \mathcal{P}, \mathcal{F} \rangle$ be any instance for MODULE MATCHING. A
 195 *canonical ordering* of H_i (w.r.t. F_i) is a total ordering over $V(H_i)$ such that the exposed
 196 vertices appear first, and every two vertices that are matched together are consecutive. In
 197 what follows, we will assume that we have access to a canonical ordering for every i . Such

¹ We stress that every polynomial function is subadditive.

198 orderings can be computed in time $\mathcal{O}(\sum_i |M_i| + |F_i|)$ by scanning all the modules and the
 199 matchings in \mathcal{F} , that is an $\mathcal{O}(\Delta m)$ provided G' has no isolated vertex.

200 Furthermore, let F be a (not necessarily maximum) matching for the subdivision $G =$
 201 $G'(\mathcal{P})$. We will make the standard assumption that, for every $v \in V(G)$, we can decide in
 202 constant-time whether v is matched by F , and if so, we can also access in constant-time to
 203 the vertex matched with v .

204 **3 A pruned modular decomposition**

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

207 ► **Definition 5.** Let $G = (V, E)$ be a graph. We call $v \in V$ a one-vertex extension if it falls
 208 in one of the following cases:

- 209 ■ $N_G[v] = V$ (*universal*) or $N_G(v) = \emptyset$ (*isolated*);
- 210 ■ $N_G[v] = V \setminus u$ (*anti-pendant*) or $N_G(v) = \{u\}$ (*pendant*), for some $u \in V \setminus v$;
- 211 ■ $N_G[v] = N_G[u]$ (*true twin*) or $N_G(v) = N_G(u)$ (*false twin*), for some $u \in V \setminus v$.

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

220 ► **Lemma 6.** *Let $G = (V, E)$ be a graph and let $v, v' \in V$ be one-vertex extensions of G . If*
 221 *v, v' are not pairwise twins then v' is a one-vertex extension of $G \setminus v$.*

222 ► **Corollary 7.** *Every graph $G = (V, E)$ has a unique pruned subgraph up to isomorphism.*

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

225 ► **Proposition 1.** For every graph $G = (V, E)$, we can compute a pruned subgraph in
 226 $\mathcal{O}(n + m \log n)$ -time.

227 **Proof.** By Corollary 7, we are left with greedily searching for, then eliminating, the one-
 228 vertex extensions. We can compute the ordered degree sequence of G in $\mathcal{O}(n + m)$ -time.
 229 Furthermore, after any vertex v is eliminated, we can update this sequence in $\mathcal{O}(|N(v)|)$ -
 230 time. Hence, up to a total update time in $\mathcal{O}(n + m)$, at any step we can detect and remove
 231 in constant-time any vertex that is either universal, isolated, pendant or anti-pendant. Fi-
 232 nally, in [20] they proposed a trie data-structure supporting the following two operations:
 233 suppression of a vertex; and detection of true or false twins (if any). The total time for all
 234 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.

235 We will term “pruned modular decomposition” of a graph G the collection of the pruned
 236 subgraphs for all the quotient subgraphs in the modular decomposition of G . Note that
 237 there is a unique pruned modular decomposition of G (up to isomorphism) and that it can
 238 be computed in $\mathcal{O}(n + m \log n)$ -time by Proposition 1 (applied to every quotient subgraph
 239 in the modular decomposition separately). Furthermore, we remark that most cases of one-
 240 vertex extensions imply the existence of non trivial modules, and so, they cannot exist in
 241 the prime quotient subgraphs of the modular decomposition. Nevertheless, such vertices
 242 may appear after removal of pendant or anti-pendant vertices (*e.g.*, in the bull graph).

243 4 Reduction rules

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

248 In Section 4.1 we recall the rules introduced in [9] for universal and isolated modules
 249 (explicitly) and for false or true twin modules (implicitly). Our main technical contributions
 250 are the reduction rules for pendant and anti-pendant modules (in Sections 4.2 and 4.3,
 251 respectively), which are surprisingly the most intricate. Finally, we end this section stating
 252 our main result (Theorem 14).

253 4.1 Simple cases

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

256 ► **Operation 1 (MATCH)**. While there are $x \in M$, $y \in N(M)$ exposed, add $\{x, y\}$ to F .

257 ► **Operation 2 (SPLIT)**. While there are $x, x' \in M$, $y, y' \in N(M)$ such that x and x' are
 258 exposed, and $\{y, y'\} \in F$, replace $\{y, y'\}$ in F by $\{x, y\}$, $\{x', y'\}$.

259 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 match-
 260 ings for H_1, H_2 , respectively. The “MATCH and SPLIT” technique consists in applying
 261 Operations 1 then 2 to $M = V(H_1)$ and $F = F_1 \cup F_2$, thereby obtaining a new matching F' ,
 262 then to $M = V(H_2)$ and $F = F'$. Based on this technique, we design the following rules:

263 ► **Reduction rule 1** (see also [9]). Suppose v_1 is isolated in G' . We set $G^* = G' \setminus v_1$,
 264 $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$, and $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Furthermore, let F^* be a maximum matching of
 265 $G^*(\mathcal{P}^*) = G[V \setminus M_1]$. We output $F^* \cup F_1$.

266 ► **Reduction rule 2** (see also [9]). Suppose v_1 is universal in G' . We set $G^* = G \setminus v_1$,
 267 $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$, $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Furthermore, let F^* be a maximum matching of the
 268 subdivision $G^*(\mathcal{P}^*) = G[V \setminus M_1]$. We apply the “MATCH and SPLIT” technique to
 269 M_1, F_1 with $V \setminus M_1, F^*$.

270 ► **Reduction rule 3**. Suppose v_1, v_2 are false twins in G' . We set $G^* = G' \setminus v_1$, $\mathcal{P}^* =$
 271 $\{H_1 \cup H_2\} \cup (\mathcal{P} \setminus \{H_1, H_2\})$, $\mathcal{F}^* = \{F_1 \cup F_2\} \cup (\mathcal{F} \setminus \{F_1, F_2\})$. We output a maximum
 272 matching of $G^*(\mathcal{P}^*) = G$.

273 ► **Reduction rule 4**. Suppose v_1, v_2 are true twins in G' . Let F_2^* be the matching of $H_1 \oplus H_2$
 274 obtained from the “MATCH and SPLIT” technique applied to M_1, F_1 with M_2, F_2 . We
 275 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
 276 a maximum matching of $G^*(\mathcal{P}^*) = G$.

277 **4.2 Anti-pendant**

278 Suppose v_1 is anti-pendant in G' . W.l.o.g., v_2 is the unique vertex that is nonadjacent to
 279 v_1 in G' . By Lemma 4, we can also assume w.l.o.g. that $E(H_i) = F_i$ for every i . In this
 280 situation, we start applying Reduction rule 1, *i.e.*, we set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \mathcal{P} \setminus \{H_1\}$,
 281 $\mathcal{F}^* = \mathcal{F} \setminus \{F_1\}$. Then, we obtain a maximum matching F^* of $G \setminus M_1$ (*i.e.*, by applying our
 282 reduction rules to this new instance). Finally, from F_1 and F^* , we compute a maximum
 283 matching F of G , using an intricate procedure. We detail this procedure next.

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

288 **► Operation 3 (REPAIR).** While there exist $x_2, y_2 \in M_2$ such that $\{x_2, y_2\} \in F_2$ and y_2
 289 is exposed in F^* , we replace any edge $\{x_2, w\} \in F^*$ by $\{x_2, y_2\}$.

290 **► Operation 4 (ATTRACT).** While there exist $x_2 \in M_2$ exposed and $\{u, w\} \in F^*$ such
 291 that $u \in N_G(M_2)$, $w \notin M_2$, we replace $\{u, w\}$ by $\{u, x_2\}$.

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

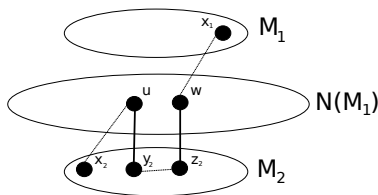
293 **► Definition 8.** A matching F of G is *good* if it satisfies the following two properties:

- 294 1. every vertex matched by $F_1 \cup F_2$ is also matched by F ;
 295 2. either every vertex in M_2 is matched, or there is no matched edge in $N_G(M_2) \times N_G(M_1)$.

296 **► Fact 1.** $F^{(0)}$ is a good matching of G .

297 **Main phase: a modified MATCH and SPLIT.** We now apply the following three
 298 operations sequentially:

- 299 1. $\text{MATCH}(M_1, F^{(0)})$ (Operation 1). Doing so, we obtain a larger good matching $F^{(1)}$.
 300 2. $\text{SPLIT}(M_1, F^{(1)})$ (Operation 2). Doing so, we obtain a larger good matching $F^{(2)}$.
 301 3. the operation UNBREAK, defined in what follows (see also Fig. 2 for an illustration):



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

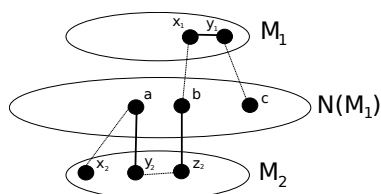
302 **► Operation 5 (UNBREAK).** While there exist $x_1 \in M_1$ and $x_2 \in M_1 \cup M_2$ exposed,
 303 and there also exist $\{y_2, z_2\} \in F_2 \setminus F^{(2)}$, we replace any two edges $\{y_2, u\}, \{z_2, w\} \in F^{(2)}$
 304 by the three edges $\{x_2, u\}, \{y_2, z_2\}$ and $\{w, x_1\}$.

305 We stress that the two edges $\{y_2, u\}, \{z_2, w\} \in F^{(2)}$ always exist since $F^{(2)}$ is a good
 306 matching of G . Furthermore doing so, we obtain a larger matching $F^{(3)}$.

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

309 **► Lemma 9.** *No vertex of M_1 can be an end in an $F^{(3)}$ -augmenting path.*

310 **Finalization phase: breaking some edges in F_1 .** Intuitively, the matching $F^{(3)}$ may
 311 not be maximum because we sometimes need to borrow some edges of F_1 in augmenting
 312 paths. So, we complete our procedure by performing the following two operations: Let U_1
 313 contain all the exposed vertices in $N(M_1)$. Consider the subgraph $G[M_1 \cup U_1] = G[M_1] \oplus$
 314 $G[U_1]$. The set U_1 is a module of this subgraph. We apply **SPLIT**($U_1, F^{(3)}$) in $G[M_1 \cup U_1]$.
 315 Doing so, we obtain a larger good matching $F^{(4)}$. Then, we apply **LOCALAUG**, defined next
 316 (see also Fig. 3 for an illustration):



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

317 ► **Operation 6** (**LOCALAUG**). While there exist $x_2 \in M_2$ and $c \in N(M_1)$ exposed, and
 318 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:

- 319 ■ we remove $\{x_1, y_1\}$ and any edge $\{a, y_2\}, \{b, z_2\}$ from $F^{(4)}$;
- 320 ■ we add $\{x_2, a\}, \{y_2, z_2\}, \{b, x_1\}$ and $\{y_1, c\}$ in $F^{(4)}$.

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

323 ► **Lemma 10.** $F^{(5)}$ is a maximum-cardinality matching of G .

324 4.3 Pendant

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

328 **First phase: greedy matching.** We apply the “**MATCH** and **SPLIT**” technique to M_1 .
 329 Doing so, we obtain a set $F_{1,2}$ of matched edges between M_1 and M_2 . We remove $V(F_{1,2})$,
 330 the set of vertices incident to an edge of $F_{1,2}$, from G . Then, there are three cases. If $M_2 \subseteq$
 331 $V(F_{1,2})$ then $M_1 \setminus V(F_{1,2})$ is isolated. We apply Reduction rule 1. If $M_1 \subseteq V(F_{1,2})$ then
 332 M_1 is already eliminated. The interesting case is when both $M_1 \setminus V(F_{1,2})$ and $M_2 \setminus V(F_{1,2})$
 333 are nonempty. In particular, suppose there remains an exposed vertex $x_1 \in M_1 \setminus V(F_{1,2})$.
 334 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
 335 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
 336 this section is that there always exists an optimal solution that contains $F_{1,2}$. This justifies
 337 a posteriori the removal of $V(F_{1,2})$ from G .

338 ► **Lemma 11.** There is a maximum matching of G that contains all edges in $F_{1,2}$.

339 We stress that during this phase, all the operations except maybe the last one increase
 340 the cardinality of the matching. Furthermore, the only possible operation that does not
 341 increase the cardinality of the matching is the replacement of an edge in F_2 by an edge
 342 in $F_{1,2}$. Doing so, either we fall in one of the two pathological cases $M_1 \subseteq V(F_{1,2})$ or
 343 $M_2 \subseteq V(F_{1,2})$ (easy to solve), or then we obtain through the replacement operation the
 344 following stronger property:

345 ► **Property 1.** All vertices in M_1 are matched by F_1 .

346 We will assume Property 1 to be true for the remaining of this section.

347 **Second phase: virtual split edges.** We complete the previous phase by performing
 348 a SPLIT between M_2, M_1 (Operation 2). That is, while there exist two exposed vertices
 349 $x_2, y_2 \in M_2$ 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
 350 the current matching. However, we encode the SPLIT operation using virtual edges in H_2 .
 351 Formally, we add a virtual edge $\{x_2, y_2\}$ in H_2 that is labeled by the corresponding edge
 352 $\{x_1, y_1\} \in F_1$. Let H_2^* and F_2^* be obtained from H_2 and F_2 by adding all the virtual edges.
 353 We set $G^* = G' \setminus v_1$, $\mathcal{P}^* = \{H_2^*\} \cup (\mathcal{P} \setminus \{H_1, H_2\})$ and $\mathcal{F}^* = \{F_2^*\} \cup (\mathcal{F} \setminus \{F_1, F_2\})$.

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

355 **Third phase: post-processing.** Let F^* be a maximum-cardinality matching of the
 356 subdivision $G^*(\mathcal{P}^*)$ (*i.e.*, obtained by applying our reduction rules to the new instance).
 357 We construct a matching F for G as follows. We add in F all the non virtual edges in F^* .
 358 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
 359 $\{x_1, y_2\}, \{x_2, y_1\}$ in F , otherwise we add $\{x_1, y_1\}$ in F . In the first case, we say that we
 360 confirm the SPLIT operation, whereas in the second case we say that we cancel it. Finally,
 361 we complete F with all the edges of F_1 that do not label any virtual edge (*i.e.*, unused
 362 during the second phase).

363 ► **Lemma 12.** *F is a maximum-cardinality matching of G .*

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

368 4.4 Main result

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

371 ► **Theorem 13** ([9]). *We can solve MODULE MATCHING for $\langle G', \mathcal{P}, \mathcal{F} \rangle$ in $\mathcal{O}(\Delta\mu \cdot p^4)$ -time.*

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

374 ► **Theorem 14.** *Let $G = (V, E)$ be a graph. Suppose that, for every prime subgraph H' in
 375 the modular decomposition of G , its pruned subgraph has order at most k . Then, we can
 376 solve MAXIMUM MATCHING for G in $\mathcal{O}(k^4 \cdot n + m \log n)$ -time.*

377 5 Applications

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

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

388 ▶ **Proposition 2.** We can solve MAXIMUM MATCHING in linear-time on graphs that can be
 389 modularly decomposed into distance-hereditary graphs and co-distance hereditary graphs.

390 Trees are a special subclass of distance-hereditary graphs. We say that a graph has
 391 *modular treewidth* at most k if every prime quotient subgraph in its modular decomposition
 392 has treewidth at most k . In particular, graphs with modular treewidth at most one are
 393 exactly the graphs that can be modularly decomposed into trees³. We stress the following
 394 consequence of Proposition 2:

395 ▶ **Corollary 15.** *We can solve MAXIMUM MATCHING in linear-time on graphs with modular-*
 396 *treewidth at most one.*

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

398 **Tree-perfect graphs.** Two graphs G_1, G_2 are P_4 -isomorphic if there exists a bijection
 399 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
 400 induces a P_4 [6]. The notion of P_4 -isomorphism plays an important role in the study of
 401 perfect graphs. A graph is *tree-perfect* if it is P_4 -isomorphic to a tree [4]. We prove the
 402 following result:

403 ▶ **Proposition 3.** Tree-perfect graphs are totally decomposable by the pruned modular
 404 decomposition. In particular, we can solve MAXIMUM MATCHING in linear-time on tree-
 405 perfect graphs.

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

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

409 ▶ **Proposition 4.** We can solve MAXIMUM MATCHING in linear-time on the graphs that
 410 can be modularly decomposed into unicycles.

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

414 6 Open problems

415 The pruned modular decomposition happens to be an interesting add up in the study of
 416 MAXIMUM MATCHING algorithms. An exhaustive study of its other algorithmic applications
 417 remains to be done. Moreover, another interesting question is to characterize the graphs that
 418 are totally decomposable by this new decomposition. We note that our pruning process can
 419 be seen as a repeated update of the modular decomposition of a graph after some specified
 420 modules (pendant, anti-pendant) are removed. However, we can only detect a restricted
 421 family of these new modules (*i.e.*, universal, isolated, twins). A fully dynamic modular
 422 decomposition algorithm could be helpful in order to further refine our framework.

423 ——— References ———

- 424 1 H.-J. Bandelt and H. Mulder. Distance-hereditary graphs. *J. of Combinatorial Theory,*
 425 *Series B*, 41(2):182–208, 1986.

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

- 426 **2** C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*,
427 43(9):842–844, 1957.
- 428 **3** J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.
- 429 **4** A. Brandstädt and V. Le. Tree-and forest-perfect graphs. *Discrete applied mathematics*,
430 95(1-3):141–162, 1999.
- 431 **5** M. Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite
432 graphs. In *ISAAC*, pages 146–155. Springer, 1996.
- 433 **6** V. Chvátal. A semi-strong perfect graph conjecture. In *North-Holland mathematics studies*,
434 volume 88, pages 279–280. Elsevier, 1984.
- 435 **7** O. Cogis and E. Thierry. Computing maximum stable sets for distance-hereditary graphs.
436 *Discrete Optimization*, 2(2):185 – 188, 2005.
- 437 **8** D. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM*
438 *Journal on Computing*, 14(4):926–934, 1985.
- 439 **9** D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes
440 of bounded clique-width graphs. In *SODA '18*, pages 2765–2784. SIAM, 2018.
- 441 **10** E. Dahlhaus and M. Karpinski. Matching and multidimensional matching in chordal and
442 strongly chordal graphs. *Discrete Applied Mathematics*, 84(1-3):79–91, 1998.
- 443 **11** G. Damiand, M. Habib, and C. Paul. A simple paradigm for graph recognition: application
444 to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–
445 111, 2001.
- 446 **12** F. Dragan. On greedy matching ordering and greedy matchable graphs. In *WG'97*, volume
447 1335 of *LNCS*, pages 184–198. Springer, 1997.
- 448 **13** S. Dubois, V. Giakoumakis, and C. El Mounir. On co-distance hereditary graphs. In *CTW*,
449 pages 94–97, 2008.
- 450 **14** G. Ducoffe and A. Popa. The use of a pruned modular decomposition for maximum match-
451 ing algorithms on some graph classes. Technical Report arXiv:1804.09407, arXiv, 2018.
- 452 **15** J. Edmonds. Paths, trees, and flowers. *Canadian J. of mathematics*, 17(3):449–467, 1965.
- 453 **16** H. Gabow and R. Tarjan. A linear-time algorithm for a special case of disjoint set union.
454 In *STOC'83*, pages 246–251. ACM, 1983.
- 455 **17** Tibor Gallai. Transitiv orientierbare graphen. *Acta Math. Hungarica*, 18(1):25–66, 1967.
- 456 **18** M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition.
457 *Computer Science Review*, 4(1):41–59, 2010.
- 458 **19** J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs.
459 *SIAM Journal on computing*, 2(4):225–231, 1973.
- 460 **20** J. Lanlignel, O. Raynaud, and E. Thierry. Pruning graphs with digital search trees. ap-
461 plication to distance hereditary graphs. In *STACS*, pages 529–541. Springer, 2000.
- 462 **21** G. Mertzios, A. Nichterlein, and R. Niedermeier. Linear-time algorithm for maximum-
463 cardinality matching on cocomparability graphs. Technical report, 2017. arXiv preprint
464 arXiv:1703.05598.
- 465 **22** S. Micali and V. Vazirani. An $O(\sqrt{VE})$ algorithm for finding maximum matching in general
466 graphs. In *FOCS'80*, pages 17–27. IEEE, 1980.
- 467 **23** D. Paulusma, F. Slivovsky, and S. Szeider. Model counting for cnf formulas of bounded
468 modular treewidth. *Algorithmica*, 76(1):168–194, 2016.
- 469 **24** M. Rao. Clique-width of graphs defined by one-vertex extensions. *Discrete Mathematics*,
470 308(24):6157–6165, 2008.
- 471 **25** M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition
472 via recursive factorizing permutations. In *ICALP*, pages 634–645. Springer, 2008.
- 473 **26** M.-S. Yu and C.-H. Yang. An $O(n)$ -time algorithm for maximum matching on cographs.
474 *Information processing letters*, 47(2):89–93, 1993.