Computing 2-homogeneous equitable partitions of graphs with a unique tree representation*

Aida Abiad[†1] and Sjanne Zeijlemaker [‡1]

[1]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

Abstract

Equitable partitions have been applied to a wide variety of topics, ranging from algebraic graph theory to clustering. An equitable partition of a graph is called *k-homogeneous* if each cell has size k. Abiad et al. showed that the existence of a 2-homogeneous equitable partition can be decided in polynomial time for cographs. In this work, we provide an alternative proof of this result, which in turn gives rise to a more general algorithm for graph classes which admit a unique tree representation. We show how the result on cographs follows from our method, and explore further applications to other graph classes.

1 Introduction

Equitable partitions are a versatile tool that have been used in many different fields of mathematics, for example for deriving sharp eigenvalue bounds on the independence number [7], constructing self-orthogonal codes [6] and clustering in various types of networks [10, 12]. A natural question is therefore, how efficiently such partitions can be computed. However, little is known about the complexity of computing equitable partitions in general [11] and the few known results focus on particular kinds of partitions or graphs. For instance, the coarsest equitable partition of a graph can be computed in polynomial time, see for example Corneil and Gotlieb [4] and Bastert [2]. Abiad et al. [1] showed that determining the existence of a 2-homogeneous equitable partition is NP-hard in general, but can be done in quadratic time for cographs. Cographs are known to have a tree representation which is unique up to isomorphism [5], a property that extends to many other graph classes. In this work, we derive a more general algorithm to compute 2-homogeneous equitable partitions in graphs that have a unique tree representation, implying the known result from [1] on cographs. Next, we explore for which other graph classes our more method can efficiently compute 2-homogeneous equitable partitions. We propose several graph classes that fall under the more general framework, and briefly discuss their structural differences and similarities with cographs.

2 Preliminaries

We consider undirected, simple and loopless graphs. A graph is denoted by $G = (V, E)$ and its number of vertices by n. The set $\{1, 2, \ldots, n\}$ is abbreviated as $[n]$. Let $G = (V, E)$ be a graph and let $\mathcal{P} = \{V_1, V_2, \ldots, V_m\}$, with $m \in [n]$, be a partition of the vertex set V. We refer to the subsets V_i

*The full version of this work will be published elsewhere.

[†]Email: a.abiad.monge@tue.nl. Research of A. A. supported by NWO (Dutch Research Council) through the grant VI.Vidi.213.085.

[‡]Email: s.zeijlemaker@tue.nl

as *cells*. A partition is called *equitable* (or *regular*) if for all i, j, each vertex in V_i has the same number of neighbors in V_j. We call a partition *k-homogeneous* if every cell has size k.

An automorphism ϕ of a graph is called an *involution* if it has order two. It is *fixed-point-free* if no vertex is mapped to itself. The following Lemma by Abiad et al. establishes a one-to-one correspondence between 2-homogeneous equitable partitions and automorphisms with the aforementioned properties.

Lemma 1 ([1, Lemma 17])**.** *Let G be a graph on n vertices. Then G has an automorphism being an involution without fixed points if and only if G admits an equitable partition with $\frac{n}{2}$ cells each having size 2.*

3 An efficient algorithm for computing 2-homogeneous equitable partitions

In general, it is NP-hard to decide whether an arbitrary graph admits a 2-homogeneous equitable partition, see [1]. Nevertheless, there exist graph classes for which the existence of a 2-homogeneous equitable partition can be established in polynomial time. Lemma 1 was also used in [1] to show that this is the case for the class of cographs (see Section 4.1 for a definition). This was done using a characterization of cographs in terms of twin classes. However, cographs can be characterized in many other ways, most notably by a tree representation which is unique up to isomorphism [5]. In this section, we derive a more general algorithm to compute 2-homogeneous equitable partitions for graphs that allow a unique tree representation, and show how the result on cographs follows from our method.

For our algorithm, we consider the following class of graphs.

Definition 2. *Let $\mathcal{G}$ be the class of all graphs satisfying the following conditions.*

(C1) There exists a polynomial-time computable rooted tree representation T which uniquely characterizes the graphs of this class up to isomorphism. Each leaf of the tree corresponds to a graph on a subset of vertices from the original graph and each internal vertex to a graph operation on the subgraphs represented by the subtrees rooted at its (unordered) children.

(C2) A fixed-point-free involution on a graph G from this class corresponds one-to-one with a label-preserving automorphism ϕ on T plus a sequence of automorphisms $\psi_1, \ldots, \psi_m$ on the subgraphs corresponding to its leaves such that

- *ϕ is an involution on the leaves of T,*
- *only maps leaf i to itself if ψ_i is a fixed-point-free involution.*

(C3) The existence of a fixed-point-free involution should be decidable in polynomial time for the subgraphs represented by the leaves.

(C4) Isomorphism of the subgraphs represented by the leaves should be decidable in polynomial time.

In (C1), (C3) and (C4), 'polynomial' means polynomial in the number of vertices of the graph.

In general, the automorphism problem on rooted labeled trees is known to be polynomial-time solvable, see Colbourn and Booth [3]. However, in the context of Lemma 1 we need to determine the existence of a particular type of automorphism which is also an involution without fixed points. In terms of the unique tree representation, this means that there should be an automorphism which swaps the leaves pairwisely or leaves them in place. However, if a leaf is not swapped, its associated subgraph must admit a fixed-point-free involution internally. In Algorithm 1, we propose a recursive procedure which determines the existence of such a mapping for labeled rooted trees representing graphs in $\mathcal{G}$. Here T_v denotes the subtree of a tree T rooted at vertex v and we define G_v to be the induced subgraph of G corresponding to the subgraph associated with leaf v in T. Since it is assumed that isomorphism can be determined efficiently for the graphs associated with the leaves of T (see Condition (C4)), we

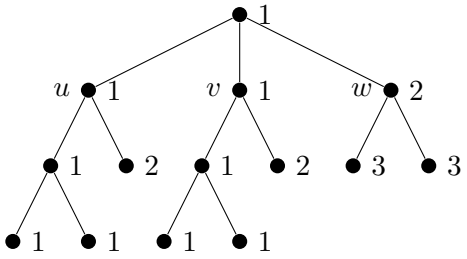

Figure 1: A j-numbered tree which admits a fixed-point-free involution on the leaves.

Algorithm 1: `hasNiceAutomorphism`

Input : A (labeled) rooted tree T representing a graph $G \in \mathcal{G}$ with root r and j-numbers assigned to each vertex, following the procedure from [3, Lemma 2.1]

Output: Does T admit an automorphism which is a fixed-point-free involution on the leaf vertices?

```
hasNiceAutomorphism(T, r)
    for each child v of r with distinct j-number do
        let k_v be the number of children with the same j-number as v
        if k_v is odd then
            if v is a leaf and G_v does not admit a fixed-point-free involution then
                return false
            else
                if hasNiceAutomorphism(T_v, v) = false then
                    return false
    return true
```

will assume that the leaves are labeled such that two leaves have the same label if and only if their associated subgraphs are isomorphic. The algorithm assumes that the input tree has been labeled using the j-numbering procedure by Colbourn and Booth [3]. These numbers are assigned in a top-down fashion to each vertex of the tree and, together with the depth of a vertex, partition the tree into its orbits under the automorphism group. A j-numbering can be computed in linear time, hence the running time of Algorithm 1 is polynomial.

Before we show correctness of Algorithm 1, we provide some intuition. Consider the rooted tree T given in Figure 1. It is clear that an automorphism of T could swap the subtrees rooted at u and v, as they are isomorphic and have the same parent. This is a fixed-point-free involution on the leaves that descend from u and v. The subtree of w cannot be mapped to another part of the graph in its entirety, but if we go one level down, we see that interchanging the children of w also results in a fixed-point-free involution on the remaining leaves.

Lemma 3. *Let $G \in \mathcal{G}$ and let T be the unique (labeled) rooted tree representing G. Algorithm 1 returns "true" if and only if T admits an automorphism which*

(i) is an involution on the leaves;

(ii) only maps a leaf to itself if the associated subgraph admits a fixed-point-free involution.

The running time of the algorithm is $O(m(m+p(n))$, where p is a polynomial and m and n denote the number of vertices of T and G respectively.

If m is polynomial in n, the running time of Algorithm 1 is also polynomial in n. Note that the number of vertices in a tree equals at most twice the number of leaves, so in order for this to hold, we only need the number of leaves of T to be polynomial in n. Combining this with Lemma 3, we obtain our main result.

Theorem 4. *Let $G \in \mathcal{G}$ be a graph with n vertices such that the number of leaves of its unique tree representation T is polynomial in n. Then the problem of deciding whether G admits an equitable partition with $\frac{n}{2}$ cells of size 2 can be solved in poly(n) time.*

Note that Algorithm 1 can easily be modified to keep track of the partial mappings and return an automorphism ψ of T satisfying conditions (i) and (ii). If we additionally compute a fixed-point-free involution for each G_v corresponding to a leaf v with odd k_v, we obtain a fixed-point-free involutionary automorphism of G, whose orbits form a 2-homogeneous equitable partition of G. Hence computing equitable partitions with $\frac{n}{2}$ cells of size 2 can also be done in polynomial time.

4 Applications

4.1 Cographs

In [1], it was shown that one can determine the existence of a 2-homogeneous equitable partition in quadratic time for the class of cographs. In this section, we provide an alternative proof of this result using Algorithm 1.

A *cograph* is a graph which does not have a P_4 as an induced subgraph. Alternatively, it can be characterized as follows.

Proposition 5 (Corneil et al. [5])**.** *A cograph is defined recursively using the following three rules.*

(i) A graph on a single vertex is a cograph.

(ii) If $G_1, \ldots, G_k$ are cographs, then so is $G_1 \cup \cdots \cup G_k$.

(iii) If G is a cograph, then so is its complement $\overline{G}$.

Note that we may equivalently replace (iii) by the condition that the join of two cographs is again a cograph. Using this characterization, the structure of a cograph can uniquely be represented by a rooted tree.

Let G be a cograph. A *cotree* of G is a rooted tree whose inner vertices each have a label 0 or 1. A leaf vertex corresponds to an induced subgraph on a single vertex and the subtree rooted at a vertex with label 0 or 1 corresponds to the union or join respectively of the subgraphs represented by its children. Note that two vertices form an edge in G if and only if their least common ancestor in the cotree has label 1. If we require the labels on a root-leaf path to be alternating, this tree is unique, see Corneil et al. [5]. The same authors observed that the graph isomorphism problem is therefore polynomial-time solvable for cographs.

Since isomorphism can be detected in polynomial time for cographs by studying the cotree, it makes sense that an automorphism of a cograph should correspond to a certain automorphism of its cotree. We make this intuition more precise in the following lemma.

Lemma 6. *Let G be a cograph with unique cotree T with alternating 0/1-labels. Then, $\phi\colon V \to V$ is an automorphism of G if and only if there exists an automorphism ψ on T such that $\psi|_V = \phi$ and which respects the 0/1-labeling.*

Lemma 6 implies that cographs satisfy Condition (C2). Combined with the uniqueness of the cotree, this implies that cographs form a subclass of $\mathcal{G}$. From Theorem 4, we then obtain an alternative proof for the following complexity result from [1].

Corollary 7 ([1, Theorem 25])**.** *The problem of deciding whether a cograph admits an equitable partition with $\frac{n}{2}$ cells of size 2 can be solved in $O(n^2)$ time.*

4.2 Other graph classes with tree representations

In the previous section, we showed how Algorithm 1 can be used to efficiently determine the existence of 2-homogeneous equitable partition in cographs. Several generalizations of cographs have been proposed in the literature, as well as other graph classes with a unique tree representation. Therefore, a natural direction for future research is to examine to which of these graphs our algorithm can be applied. Below, we highlight three promising graph classes, and indicate which further steps need to be taken to apply our approach.

Tree-cographs As we have seen in Proposition 5, cographs can be defined recursively by starting with a set of isolated vertices and repeatedly applying disjoint union and complementation operations. A *tree-cograph* is a graph which can be obtained by applying the same operations to a set of trees. Tree-cographs generalize both the class of trees and cographs. It was shown by Tinhofer [13] that these graphs can be uniquely characterized by a tree representation. Contrary to cographs, the inner vertices of this tree represent the disjoint union and complementation operations and the leaves each correspond to a tree. It is easy to see that this class of graphs satisfies Conditions (C1), (C3) and (C4). However, as the proof of Lemma 6 makes use of the properties of the join operator and a characterization of adjacency in cographs, an alternative proof is needed to show that tree-cographs are a subclass of $\mathcal{G}$.

P_4-sparse graphs Cographs are the class of graphs which contain no induced P_4. A natural generalization of this concept is the class of P_4-sparse graphs. A graph is *P_4-sparse* if every set of five vertices induces at most one P_4. Jamison and Olariu [8] showed that a graph is P_4-sparse if and only if it can be constructed from a set of isolated vertices through applying the disjoint union, join and a third operator denoted by ② (see [8] for more details). Moreover, this constructive characterization gives rise to a unique tree representation. Once again, an alternative to Lemma 6 is needed to show inclusion in $\mathcal{G}$, as Condition (C2) is not trivially satisfied.

Interval graphs An *interval graph* is the intersection graph of a set of intervals on the line. Interval graphs can be represented by a *PQ-tree* [9], whose leaves correspond to the maximal cliques of the associated graph. Colbourn and Booth [3] proved that an automorphism of an interval graph correspond one-to-one with an automorphism of its PQ-tree and a certain permutation of the vertices. This gives us a direct link between the fixed-point-free involutions of an interval graph and automorphisms of its PQ-tree, as needed for Condition (C2). However, to apply Algorithm 1, two additional properties of the PQ-tree need to be taken into consideration. Firstly, some vertices of the PQ-tree have ordered children which may not be exchanged arbitrarily. These restrictions can easily be incorporated into the algorithm. Secondly, the maximal cliques of an interval graph, and hence the subgraphs associated with the leaves of the PQ-tree, may not be disjoint. Algorithm 1 traverses a given tree in a top-down manner and attempts to pair up each subtree individually to obtain a fixed-point-free automorphism. For PQ-trees, consistency needs to be ensured for vertices appearing in multiple subtrees.

References

[1] A. Abiad, C. Hojny, and S. Zeijlemaker. Characterizing and computing weight-equitable partitions of graphs. *Linear Algebra and its Applications*, 645:30–51, 2022.

[2] O. Bastert. Computing equitable partitions of graphs. *Match*, 40:265-–272, 1999.

[3] C. J. Colbourn and K. S. Booth. Linear time automorphism algorithms for trees, interval graphs, and planar graphs. *SIAM Journal on Computing*, 10(1):203–225, 1981.

[4] D. G. Corneil and C. C. Gotlieb. An efficient algorithm for graph isomorphism *Journal of the ACM*, 17(1):51–64, 1970.

[5] D. G. Corneil, H. Lerchs, and L. S. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.

[6] D. Crnković, S. Rukavina, and A. Švob. Self-orthogonal codes from equitable partitions of association schemes. *Journal of Algebraic Combinatorics*, 55:157–171, 2022.

[7] A. J. Hoffman. On eigenvalues and colorings of graphs. In *Selected Papers of Alan J Hoffman: With Commentary*, 407-–419. World Scientific, 2003.

[8] B. Jamison and S. Olariu. A tree representation for P_4-sparse graphs. *Discrete Applied Mathematics*, 35(2):115–129, 1992.

[9] G. S. Lueker and K. S. Booth. A linear time algorithm for deciding interval graph isomorphism. *Journal of the ACM*, 26(2):183–195, 1979.

[10] M. Mattioni and S. Monaco. Cluster partitioning of heterogeneous multi-agent systems. *Automatica*, 138: 110136, 2022.

[11] B. D. McKay. Complexity of equitable partitions. URL: `https://mathoverflow.net/q/96858` (version: 2012-05-14)

[12] B. Prasse, K. Devriendt, and P. Van Mieghem. Clustering for epidemics on networks: a geometric approach. *Chaos: an Interdisciplinary Journal of Nonlinear Science*, 31(6):063115, 2021.

[13] G. Tinhofer. Strong tree-cographs are Birkhoff graphs. *Discrete Applied Mathematics*, 22(3):275–288, 1988.