Independent Set Reconfiguration in Cographs

Paul Bonsma(✉)

Faculty of EEMCS, University of Twente, PO Box 217,
7500 AE Enschede, The Netherlands
p.s.bonsma@ewi.utwente.nl

Abstract. We study the following independent set reconfiguration problem: given two independent sets I and J of a graph G, both of size at least k, is it possible to transform I into J by adding and removing vertices one-by-one, while maintaining an independent set of size at least k throughout? This problem is known to be PSPACE-hard in general. For the case that G is a cograph on n vertices, we show that it can be solved in polynomial time. More generally, we show that for a graph class $\mathcal{G}$ that includes all chordal and claw-free graphs, the problem can be solved in polynomial time for graphs that can be obtained from a collection of graphs from $\mathcal{G}$ using disjoint union and complete join operations.

1 Introduction

Reconfiguration problems have been studied often in recent years. These arise in settings where the goal is to transform feasible solutions to a problem in a step-by-step manner, while maintaining a feasible solution throughout. A *reconfiguration problem* is obtained by defining *feasible solutions* (or configurations) for *instances* of the problem, and a (symmetric) *adjacency relation* between solutions. This defines a *solution graph* for every instance, which is usually exponentially large in the input size. Usually, it is assumed that *adjacency* and *being a feasible solution* can be tested in polynomial time. Typical questions that are studied are deciding the existence of a path between two given solutions *(reachability)*, finding shortest paths between solutions, deciding whether the solution graph is connected or giving sufficient conditions for this, and giving bounds on its diameter. For example, the literature contains such results on the reconfiguration of vertex colorings [1,3,7,9–11], boolean assignments that satisfy a given formula [16], independent sets [17,20,22,24], matchings [20], shortest paths [4,5,21], subsets of a (multi-)set of integers [14,19], etc. Techniques for many different reconfiguration problems are discussed in [20,24]. See the recent survey by Van den Heuvel [18] for an overview of and introduction to reconfiguration problems, and a discussion of their various applications.

One of the most well-studied problems of this kind is the reconfiguration of *independent sets* (which are sets of pairwise nonadjacent vertices). For a graph G

Supported by the European Community's Seventh Framework Programme (FP7/2007–2013), grant agreement n° 317662.

and integer k, the independent sets of size at least/exactly k of G form the feasible solutions. Independent sets are also called *token configurations*, where the independent set vertices are viewed as *tokens.* Three types of adjacency relations have been studied in the literature: in the *token jumping (TJ)* model [20,22], a token can be moved from any vertex to any other vertex. In the *token sliding (TS)* model, tokens can be moved along edges of the graph [17,22]. In the *token addition and removal (TAR)* model [20,22], tokens can be removed and added in arbitrary order, though at least k tokens should remain at any time (k is the *token lower bound*). Of course, in all of these cases, an independent set should be maintained.

The *reachability problem* has received the most attention in this context: given two independent sets I and J of a graph G, and possibly a token lower bound $k \leq \min\{|I|, |J|\}$, is there a path (or *reconfiguration sequence*) from I to J in the solution graph? We call this problem *TJ-Reachability, TS-Reachability* or *TAR-Reachability*, depending on the adjacency relation that is used. Kamiński et al. [22] showed that the TAR-Reachability problem generalizes the TJ-Reachability problem. For all three adjacency relations, this problem is PSPACE-hard, even in perfect graphs [22], and even in planar graphs of maximum degree 3 [17] (see also [7]). In [20] an alternative, simple PSPACE-hardness proof is given. In addition, in [22], the problem of deciding whether there exists a path of length at most l between two solutions is shown to be strongly NP-hard, for all three adjacency models.

On the positive side, these problems can be solved in polynomial time for various restricted graph classes. The result on matching reconfiguration by Ito et al. [20] implies that for line graphs, TJ-Reachability and TAR-Reachability can be solved efficiently. This result has recently been generalized to claw-free graphs, also for TS-Reachability [8]. Kamiński et al. [22] give an efficient algorithm for TS-Reachability in cographs, and show that for TJ-Reachability in even-hole-free graphs, a reconfiguration sequence of length $|I \backslash J|$ exists between every pair of independent sets I and J. TAR-Reachability has also been studied under the name *Vertex Cover Reconfiguration* in [24], where parameterized complexity results for the problem are given. (Recall that I is an independent set of G if and only if $V(G) \backslash I$ is a vertex cover of G.)

New results and techniques. In this paper, we show that TAR-Reachability can be solved in polynomial time for cographs. Using [22], it follows that the same holds for TJ-Reachability. This answers an open question from [22]. Recall that a graph is a *cograph* iff it has no induced path on four vertices. Alternatively, cographs can be defined as graphs that can be obtained from a collection of trivial (one vertex) graphs by repeatedly applying *(disjoint) union* and *(complete) join* operations. The order of these operations can be described using a rooted *cotree.* This characterization allows efficient dynamic programming (DP) algorithms for various NP-hard problems. Our algorithm is also a DP algorithm over the cotree, albeit more complex than many known DP algorithms on cographs. For both solutions I and J, certain values are computed, using first a *bottom up* DP phase, and next a *top down* DP phase over the cotree. Using these values, we can conclude whether J is reachable from I. Because of this method, we in fact obtain a stronger result: TJ- and TAR-Reachability can be decided efficiently for any graph that can be obtained using join and union operations, when starting with a collection of base graphs from a graph class $\mathcal{G}$ that satisfies the following properties: (i) For any graph in $\mathcal{G}$, the TAR-Reachability problem can be decided efficiently, and (ii) for any graph in $\mathcal{G}$ and independent set I, the size of a maximum independent set that is TAR-reachable from I can be computed efficiently, for all token lower bounds $k \leq |I|$. Results from [8,15,22,23,25] can easily be combined to show that chordal graphs and claw-free graphs satisfy these properties. In all, this yields quite a rich graph class for which this PSPACE-hard problem can be solved efficiently. Considering the fact that TAR-Reachability is PSPACE-hard for perfect graphs [22], the boundary between hard and easy graph classes for this problem starts to become clear.

This paper presents one of the first nontrivial examples of how dynamic programming over graph decompositions can be used to solve reconfiguration problems. (We remark that a DP approach has also been used to show that the PSPACE-hard Shortest Path Reconfiguration problem can be solved in polynomial time on planar graphs [4], using a problem-specific layer decomposition of the graph.) This is especially interesting since cographs form the base class for various graph width measures: cographs are exactly the graphs of cliquewidth at most two, and exactly the graphs of modular-width two [13]. We expect that our method forms a first step towards efficiently solving various reconfiguration problems for graphs of bounded modular-width, and provides useful concepts for addressing other graph classes/decompositions. However, for graphs of bound cliquewidth, similar efficient algorithms should not be expected, since it was shown very recently that many reconfiguration problems, including TAR-Reachability and TJ-Reachability, remain PSPACE-hard for graphs of bandwidth/treewidth/cliquewidth at most k, for some constant k [27].

Our DP algorithm for the TAR-Reachability problem is presented in Sects. 3–5. First, in Sect. 3, an example is given, the proof of this statement is outlined, and a detailed overview of Sects. 4 and 5 is given. In Sect. 6, examples of graph classes are given to which this algorithm applies. We start in Sect. 2 with precise definitions, and end in Sect. 7 with a discussion. Statements for which additional proof details can be found in the full version of this paper [6] are marked with a star.

2 Preliminaries

By $\alpha(G)$ we denote the maximum size of an independent set in G. In this paper, we use the *token addition and removal (TAR)* model for independent set reconfiguration. For a graph G and integer k, the vertex set of the graph $\text{TAR}_k(G)$ is the set of all independent sets of size at least k in G. Two distinct independent sets I and J are adjacent in $\text{TAR}_k(G)$ if there exists a vertex $v \in V(G)$ such that $I \cup \{v\} = J$ or $I = J \cup \{v\}$. Vertices from independent sets will also be called *tokens*, and we will also say that J is obtained from I by *adding one token on v* resp. *removing one token from v*.

For an integer k and two independent sets I and J of G with $|I| \geq k$ and $|J| \geq k$, we write $I \leftrightarrow_k^G J$ if $\mathrm{TAR}_k(G)$ contains a walk from I to J. Such a walk in $\mathrm{TAR}_k(G)$ (a sequence of independent sets) is also called a k-TAR-sequence *for G from I to J*. To avoid discussing trivial cases in our proofs, we allow that a k-TAR-sequence contains consecutive sets that are identical. Observe that $I \leftrightarrow_0^G J$ always holds, and that the relation $\leftrightarrow_k^G$ is an equivalence relation, for all G and k. The superscript G is omitted if the graph in question is clear. If G and k are clear from the context, we will also simply say that *J is reachable from I*.

A *generalized cotree* is a binary tree T with root r, together with

- a partition of the nonleaf vertices into *union nodes* and *join nodes*, and
- a graph G_u for every leaf u of T, such that for any two leaves u and v, the graphs G_u and G_v are vertex and edge disjoint.

Vertices of T are called *nodes*. With every node $u \in V(T)$ we associate a graph G_u in the following way: for leaves u, G_u is as given. Otherwise, u has two child nodes; denote these by v and w. If u is a union node, then G_u is the *disjoint union* of G_v and G_w. If u is a join node, then G_u is obtained by taking the *complete join* of G_v and G_w. This operation is defined as follows: start with the disjoint union of G_v and G_w, and add edges yz for every combination of $y \in V(G_v)$ and $z \in V(G_w)$. For a node $u \in V(T)$, we denote $V_u = V(G_u)$. A generalized cotree T is called a *cotree* if for every leaf $v \in V(T)$, the graph G_v consists of a single vertex. Such a leaf is called a *trivial leaf*. (See Fig. 1(d) for an example.)

Let T be a (generalized) cotree, with root r. For a graph G, we say that T is a *(generalized) cotree for* G if $G_r = G$. A graph G is called a *cograph* if there exists a cotree for G. Let $\mathcal{G}$ be a graph class. We say that a generalized cotree T for a graph G is a *cotree decomposition of G into $\mathcal{G}$-graphs* if for every leaf $v \in V(T)$, the graph $G_v \in \mathcal{G}$.

3 Example and Proof Outline

In Fig. 1, three independent sets A, B and C are shown for a cograph G. In order to go from A to B in $\mathrm{TAR}_5(G)$, an independent set must be visited which has no tokens on the component G_x, and therefore at least five tokens on the other two components. The only such independent set of G is C. Using similar observations, it can be verified that the *shortest* 5-TAR-sequence from A (or B) to C is unique up to symmetries, and has length twelve (six additions and deletions). Hence the shortest 5-TAR-sequence from A to B has length 24. In general, deciding whether $A \leftrightarrow_k^G B$ requires computing the following values $\lambda_k^I(v)$, which indicate the minimum number of vertices of V_v that must be contained in any independent set reachable from I.

Definition 1. *Let T be a generalized cotree for a graph G, I be an independent set of G, and $k \leq |I|$. For $v \in V(T)$, define $\lambda_k^I(v) = \min |J \cap V_v|$ over all independent sets J of G with $I \leftrightarrow_k^G J$.*

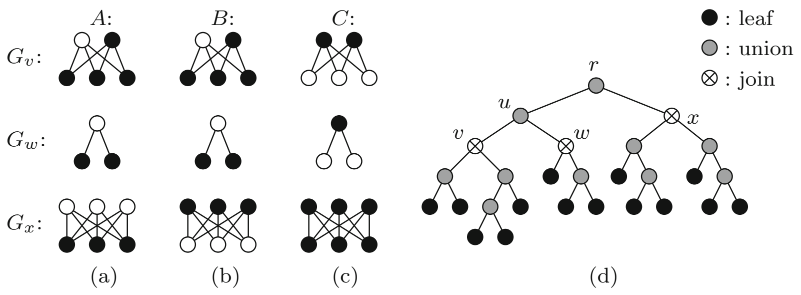

Fig. 1. (a), (b), (c): A cograph G, with independent sets A, B and C indicated by the white vertices. Any 5-TAR-sequence from A to B must visit C and use all vertices of G. (d): A cotree of G with root r, with join nodes v, w and x corresponding to the components of G.

For instance, in the example from Fig. 1, $\lambda_5^A(x) = 0 = \lambda_5^B(x)$, and this fact is essential for concluding that $A \leftrightarrow_5^G B$ in this case. The following theorem characterizes whether B is reachable from A, using the values from Definition 1.

Theorem 2. *Let T be a generalized cotree for a graph G. Let A and B be two independent sets of G of size at least k. Then $A \leftrightarrow_k^G B$ if and only if*

1. *for all nodes $u \in V(T)$, $\lambda_k^A(u) = \lambda_k^B(u)$, and*
2. *for all leaves $u \in V(T)$, $(A \cap V_u) \leftrightarrow_\ell^{G_u} (B \cap V_u)$, where $\ell = \lambda_k^A(u)$.*

The forward direction of the proof is straightforward: if $A \leftrightarrow_k^G B$, then since $\leftrightarrow_k^G$ is an equivalence relation, any independent set J is reachable from A if and only if it is reachable from B. It follows that $\lambda_k^A(v) = \lambda_k^B(v)$ for all $v \in V(T)$. The second property follows by restricting all independent sets in a k-TAR-sequence from A to B to the subgraph G_u for any leaf $u \in V(T)$. By definition, these all have size at least $\ell = \lambda_k^A(u)$, so this yields an ℓ-TAR-sequence from $A \cap V_u$ to $B \cap V_u$ for G_u. In Sect. 5, the backward direction of the proof is given.

In order to efficiently decide whether $A \leftrightarrow_k^G B$, it remains to compute the values $\lambda_k^I(v)$ for all $v \in V(T)$ and $I = A, B$. In the example from Fig. 1, it holds that $\lambda_5^A(x) = 0$. This is because on the subgraph G_u, which is the disjoint union of components G_v and G_w (see Fig. 1(d)), it is possible to reconfigure from the initial independent set A to an independent set with at least five tokens on G_u, while keeping at least two tokens on G_u throughout. This indicates that in order to compute the values $\lambda_k^I(v)$, the following values must be computed.

Definition 3. *Let T be a generalized cotree for G, and let I be an independent set of G. For $v \in V(T)$ and $\ell \in \{0, \dots, |I \cap V_v|\}$, denote by $\mu_\ell^I(v)$ the maximum of $|J|$ over all independent sets J of G_v with $(I \cap V_v) \leftrightarrow_\ell^{G_v} J$.*

Note that $\mu_0^I(v) = \alpha(G_v)$ (regardless of the choice of I). The value $\mu_\ell^I(v)$ depends only on the situation in the subgraph G_v; not on the entire graph. This is in contrast to the values $\lambda_k^I(v)$. So the values $\mu_\ell^I(u)$ for a node u with children v and

w can be computed using only the values $\mu^I_{\ell'}(v)$ and $\mu^I_{\ell'}(w)$ for different choices of ℓ', so using a *bottom up* dynamic programming algorithm. This can then be used to compute values $\lambda^I_k(u)$, which requires considering the entire graph, so this uses a *top down* dynamic programming algorithm. The DP rules are given in Sect. 4. Together with Theorem 2, this yields our main algorithmic result, given in the next theorem, which is also proved in Sect. 5.

Theorem 4. *Let T be a generalized cotree for a graph G on n vertices, let $k \in \mathbb{N}$ and let A and B be independent sets of G. If for every nontrivial leaf $v \in V(T)$ and relevant integer ℓ, (1) the values $\mu^A_\ell(v)$ and $\mu^B_\ell(v)$ are known, and (2) it is known whether $(A \cap V_v) \leftrightarrow^{G_v}_\ell (B \cap V_v)$, then in polynomial time it can be decided whether $A \leftrightarrow^G_k B$.*

In particular, Theorem 4 implies that for any two independent sets A and B for a *cograph* G, it can be decided in polynomial time whether $A \leftrightarrow^G_k B$.

For all of our proofs, an essential (easy to see) fact is that for every node u, the vertex set V_u is a module of G. A *module* of a graph G is a set $M \subseteq V(G)$ such that for every $v \in V(G)\backslash M$, either $M \subseteq N(v)$ or $M \cap N(v) = \emptyset$. Note that we will also consider $V(G)$ to be a (trivial) module of G. Modules are very useful for independent set reconfiguration, since to some extent, we can reconfigure within the module and outside of the module independently; only the *number* of tokens on the module matters. The following two lemmas make this more precise, and present two useful properties for the proofs below.

Lemma 5. *(*) Let M be a module of a graph G, let k and ℓ be integers, and let A be an independent set of G, with $|A \cap M| \geq \max\{1, \ell\}$ and $|A| \geq k$. Denote $H = G[M]$. If there exists an independent set B of G with $A \leftrightarrow^G_k B$ and $|B \cap M| \leq \ell$, and if there exists an independent set C of H with $(A \cap M) \leftrightarrow^H_\ell C$, then there exists an independent set D of G with $A \leftrightarrow^G_k D$ and $D \cap M = C$.*

Lemma 6. *(*) Let M be a module of a graph G, such that M can be partitioned into two sets M_1 and M_2 with no edges between M_1 and M_2. Let A be an independent set of G, let B_1 be an independent set of G with $A \leftrightarrow^G_k B_1$, that maximizes $|B_1 \cap M_1|$ among all such sets, and let B_2 be an independent set of G with $A \leftrightarrow^G_k B_2$. Then there exists an independent set C of G with $A \leftrightarrow^G_k C$ and $C \cap M_i = B_i \cap M_i$ for $i \in \{1, 2\}$.*

4 Dynamic Programming Rules

Throughout this section, T denotes a generalized cotree of G and I denotes an independent set of G. We first show how to compute the values $\mu^I_\ell(u)$ for every type of node u. For trivial leaf nodes, this is easy.

Proposition 7. *Let $u \in V(T)$ be a trivial leaf node. Then $\mu^I_\ell(u) = 1$ for all ℓ.*

For join nodes u, the computation of $\mu^I_\ell(u)$ is still relatively straightforward. Note that for any independent set I, u has a child w with $V_w \cap I = \emptyset$.

Proposition 8. *Let $u \in V(T)$ be a join node. Let w be a child of u with $I \cap V_v = \emptyset$, and let v be the other child of u. Then $\mu^I_\ell(u) = \mu^I_\ell(v)$ for all $\ell \geq 1$, and $\mu^I_0(u) = \max\{\mu^I_0(v), \mu^I_0(w)\}$.*

Proof: Because all edges are present between G_v and G_w, a maximum independent set of G_u is either a maximum independent set of G_v or of G_w, so $\mu^I_0(u) = \alpha(G_u) = \max\{\alpha(G_v), \alpha(G_w)\} = \max\{\mu^I_0(v), \mu^I_0(w)\}$. Now consider the case $\ell \geq 1$, and thus $|I \cap V_u| \geq 1$. Then initially all tokens of I are on the child G_v. As long as there is at least one token on G_v, no tokens can be added to G_w. So essentially, G_w can be ignored, and thus $\mu^I_\ell(u) = \mu^I_\ell(v)$. □

For union nodes u with children v and w, computing the values $\mu^I_\ell(u)$ is more complicated, and requires studying ℓ-TAR-sequences for G_u of the following type. Let $x_0 = I \cap V_v$. Observe that from the initial independent set $I \cap V_u$ we can reach an independent set with $\mu^I_{x_0}(v)$ tokens on V_v, and $y_0 := \max\{0, \ell - \mu^I_{x_0}(v)\}$ tokens on V_w, while keeping at least ℓ tokens on V_u throughout. Call this an independent set of *type* $(\mu^I_{x_0}(v), y_0)$. From this, we can subsequently reach an independent set of type $(x_1, \mu^I_{y_0}(w))$, with $x_1 := \max\{0, \ell - \mu^I_{y_0}(w)\}$. Next, an independent set of type $(\mu^I_{x_1}(v), y_1)$ with $y_1 := \max\{0, \ell - \mu^I_{x_1}(v)\}$ can be reached, etc. This process continues with finding ever lower x- and y-values, until a 'stable tuple' (x, y) is obtained. This motivates the following definition.

Definition 9. *For a union node $u \in V(T)$ with left child v and right child w, and integer $\ell \leq |I \cap V_u|$, call a tuple (x, y) of integers with $x \leq |I \cap V_v|$ and $y \leq |I \cap V_w|$* ℓ-stable *if $x = \max\{0, \ell - \mu^I_y(w)\}$ and $y = \max\{0, \ell - \mu^I_x(v)\}$. Call an ℓ-stable tuple (x, y)* maximum *if there is no ℓ-stable tuple (x', y') with $x' \geq x$, $y' \geq y$ and $(x, y) \neq (x', y')$.*

It can be shown that there is a unique maximum ℓ-stable tuple, which can be characterized as follows. Using this characterization, Lemma 11 shows how the values $\mu^I_\ell(u)$ can be computed for a join node u.

Lemma 10. *(*) Let $u \in V(T)$ be a union node, with left child v and right child w. For $\ell \in \{0, \dots, |I \cap V_u|\}$, let $x = \min |J \cap V_v|$ and $y = \min |J \cap V_w|$, where in both cases the minimum is taken over all independent sets J of G_u with $(I \cap V_u) \leftrightarrow^{G_u}_\ell J$. Then (x, y) is the unique maximum ℓ-stable tuple for I and u.*

Lemma 11. *(*) Let $u \in V(T)$ be a union node, with left child v and right child w. For $\ell \in \{0, \dots, |I \cap V_u|\}$, let (x, y) be the unique maximum ℓ-stable tuple for I and u. Then $\mu^I_\ell(u) = \mu^I_x(v) + \mu^I_y(w)$.*

We will now show how the values $\lambda^I_k(v)$ can be computed for all nodes $v \in V(T)$. For the case that v is a union node, this requires knowledge of the unique maximum ℓ-stable tuple. For the root node of T, the value is trivial.

Proposition 12. *Let r be the root node of T. Then $\lambda^I_k(r) = k$.*

Proposition 13. *Let $u \in V(T)$ be a join node, with children v and w such that $I \cap V_w = \emptyset$. Then $\lambda^I_k(v) = \lambda^I_k(u)$ and $\lambda^I_k(w) = 0$.*

Proof: Considering I, $\lambda_k^I(w) = 0$ follows immediately. If $\lambda_k^I(u) = 0$, then obviously $\lambda_k^I(v) = 0$. Adding a token to G_w requires first reaching an independent set with no tokens on G_v, and thus requires $\lambda_k^I(u) = 0$. So if $\lambda_k^I(u) \geq 1$, then G_w can essentially be ignored, and therefore $\lambda_k^I(v) = \lambda_k^I(u)$ in that case. □

Lemma 14. *Let $u \in V(T)$ be a union node, with left child v and right child w. Let $\ell = \lambda_k^I(u)$, and let (x, y) be the maximum ℓ-stable tuple for I and u. Then $\lambda_k^I(v) = x$ and $\lambda_k^I(w) = y$.*

Proof: Denote $I_u = I \cap V_u$. We first show that $\lambda_k^I(v) \geq x$ and $\lambda_k^I(w) \geq y$. Consider a k-TAR-sequence $I_0, \dots, I_p$ for G with $I_0 = I$ and $|I_p \cap V_v| = \lambda_k^I(v)$. For every i, denote $I'_i = I_i \cap V_u$, and consider the sequence $I'_0, \dots, I'_p$. By definition of $\ell = \lambda_k^I(u)$, for every i it holds that $|I'_i| \geq \ell$, so $I_u \leftrightarrow_\ell^{G_u} I'_p$. Using Lemma 10 it then follows that $\lambda_k^I(v) = |I_p \cap V_v| \geq x$. Analogously, $\lambda_k^I(w) \geq y$ follows.

We will now prove that $\lambda_k^I(v) \leq x$ and $\lambda_k^I(w) \leq y$. The case $\ell = 0$ is obvious, so assume $\ell \geq 1$. By Lemma 10, there exist independent sets J_1 and J_2 of G_u with $I_u \leftrightarrow_\ell^{G_u} J_1$, $I_u \leftrightarrow_\ell^{G_u} J_2$, $|J_1 \cap V_v| = x$ and $|J_2 \cap V_w| = y$. By the definition of $\ell = \lambda_k^I(u)$, there exists an independent set B of G with $I \leftrightarrow_k^G B$ and $|B \cap V_u| = \ell$. We can now apply Lemma 5 twice, with V_u and I in the role of M and A, and J_1 or J_2 respectively in the role of C, to conclude that there exist independent sets D_1 and D_2 of G with $I \leftrightarrow_k^G D_1$, $I \leftrightarrow_k^G D_2$, $D_1 \cap V_u = J_1$ and $D_2 \cap V_u = J_2$. So $|D_1 \cap V_v| = x$ and $|D_2 \cap V_w| = y$, and thus $\lambda_k^I(v) \leq x$ and $\lambda_k^I(w) \leq y$. □

5 Algorithm Summary and Main Theorems

We can now prove our two main theorems; first we prove our characterization of $A \leftrightarrow_k^G B$ in terms of the values $\lambda_k^I(u)$, and secondly we summarize how these values can be computed efficiently.

Proof of Theorem 2: The forward direction of the proof was given in Sect. 3. We now prove the backward direction. Assume that the two properties given in the theorem statement hold. So we may denote $\lambda_k(u) = \lambda_k^A(u) = \lambda_k^B(u)$ for all nodes u. We prove the following claim by induction over T:

Claim A: For all nodes $u \in V(T)$: $(A \cap V_u) \leftrightarrow_{\lambda_k(u)}^{G_u} (B \cap V_u)$.

For leaf nodes $u \in V(T)$ (induction base), the statement follows immediately from the second property. To prove the induction step, first consider a join node $u \in V(T)$ with children v and w. Suppose that $\lambda_k(v) \geq 1$. This implies $A \cap V_v \neq \emptyset$ and $B \cap V_v \neq \emptyset$. Therefore, since u is a join node, $A \cap V_u = A \cap V_v$ and $B \cap V_u = B \cap V_v$. In addition, $\lambda_k(u) = \lambda_k(v)$ (Proposition 13). From these facts, and the induction assumption $(A \cap V_v) \leftrightarrow_{\lambda_k(v)}^{G_v} (B \cap V_v)$, we conclude that $(A \cap V_u) \leftrightarrow_{\lambda_k(u)}^{G_u} (B \cap V_u)$. The case $\lambda_k(w) \geq 1$ is analog. On the other hand, if $\lambda_k(v) = \lambda_k(w) = 0$, then $\lambda_k(u) = 0$ (Proposition 13). Claim A follows for u since $(A \cap V_u) \leftrightarrow_0^{G_u} (B \cap V_u)$ trivially holds.

Next, consider the case that $u \in V(T)$ is a union node with left child v and right child w. Denote $\ell = \lambda_k(u)$, $x = \lambda_k(v)$ and $y = \lambda_k(w)$. By Lemma 14, (x, y) is the maximum ℓ-stable tuple for u, for both A and B. We define C_v to be an independent set of G_v with $(A \cap V_v) \leftrightarrow_x^{G_v} C_v$, with maximum size among all such sets, and define C_w to be an independent set of G_w with $(A \cap V_w) \leftrightarrow_y^{G_w} C_w$, with maximum size among all such sets. By induction, $(A \cap V_v) \leftrightarrow_x^{G_v} (B \cap V_v)$ holds, so it also holds that $(B \cap V_v) \leftrightarrow_x^{G_v} C_v$, and that C_v has maximum size among all such reachable sets. Analogously, $(B \cap V_w) \leftrightarrow_y^{G_w} C_w$ holds, and C_w has maximum size among all such reachable sets. Define $C_u = C_v \cup C_w$. We will now show that C_u is reachable from both $A \cap V_u$ and $B \cap V_u$, which proves Claim A for node u.

Lemma 10 shows that there exists an independent set J of G_u with $(A \cap V_u) \leftrightarrow_\ell^{G_u} J$ and $|J \cap V_v| = x$. Using this, we argue that there exists an independent set J_1 of G_u with $(A \cap V_u) \leftrightarrow_\ell^{G_u} J_1$ and $J_1 \cap V_v = C_v$. If $A \cap V_v = \emptyset$, then this claim is trivial. Otherwise, we can apply (module) Lemma 5 to draw this conclusion (using V_v, G_u, J and C_v in the roles of the module M, entire graph G, and independent sets B and C, respectively). Analogously, we may conclude that there exists an independent set J_2 of G_u with $(A \cap V_u) \leftrightarrow_\ell^{G_u} J_2$ and $J_2 \cap V_w = C_w$. Since $C_u = C_v \cup C_w$, we can now apply (module) Lemma 6 (with G_u in the role of the entire graph, V_v and V_w in the roles of disjoint modules M_1 and M_2, and J_1 and J_2 in the roles of B_1 and B_2), to conclude that $A \cap V_u \leftrightarrow_\ell^{G_u} C_u$. For this, we require the fact that C_v has maximum size among all independent sets of G_v that are reachable from $A \cap V_v$.

The argument from the previous paragraph also holds when replacing A by B, since C_v and C_w are also maximum reachable independent sets from $B \cap V_v$ and $B \cap V_w$. So $B \cap V_u \leftrightarrow_\ell^{G_u} C_u$ also holds. Hence $A \cap V_u \leftrightarrow_\ell^{G_u} C_u \leftrightarrow_\ell^{G_u} B \cap V_u$, which proves Claim A for u.

This concludes the induction proof of Claim A. Applying Claim A to the root node r of T shows that $A \leftrightarrow_k^G B$, since $\lambda_k(r) = k$ (Proposition 12), and $G = G_r$, and therefore concludes the proof of the theorem. □

Proof of Theorem 4: First we use a *bottom up* dynamic programming algorithm, to compute the values $\mu_\ell^A(u)$ and $\mu_\ell^B(u)$ for every node u and relevant integer ℓ, and to compute the maximum ℓ-stable tuples for every union node and relevant integer ℓ. This can be done in polynomial time using the rules given in Proposition 7, Proposition 8 and Lemma 11. Note that maximum ℓ-stable tuples can easily be computed in polynomial time by testing a quadratic number of possible tuples (Definition 9).

Next, we start the *top down* phase of the dynamic programming algorithm, where we compute the values $\lambda_k^A(u)$ and $\lambda_k^B(u)$ for every node u. This can be done in polynomial time using the rules given in Propositions 12 and 13, and Lemma 14. Note that applying Lemma 14 to a union node u requires the previously computed maximum ℓ-stable tuple (x, y) for $I = A, B$, with $\ell = \lambda_k^I(u)$. This is why the bottom up phase is required. At this point, the characterization given in Theorem 2 can be used to conclude whether $A \leftrightarrow_k^G B$. □

6 Examples of Suitable Graph Classes

Consider T, G, A, B and k as in Theorem 4. If $v \in V(T)$ is a trivial leaf, then for every relevant value ℓ, $\mu_\ell^A = 1$ and $\mu_\ell^B = 1$ hold (Proposition 7), and clearly $(A \cap V_v) \leftrightarrow_\ell^{G_v} (B \cap V_v)$ holds. So combined with the fact that a cotree of a cograph G can be found in linear time [12], Theorem 4 implies that the TAR-Reachability problem can be decided in polynomial time for a cograph G.

Theorem 4 is however much stronger, and implies that TAR-Reachability can be decided efficiently for much richer graph classes. Recall that a (simple) graph G is *chordal*/*even-hole-free*/*claw-free* if it does not contain as an *induced subgraph* a cycle of length at least four/a cycle of even length/a $K_{1,3}$, respectively. By applying independent set reconfiguration results from [22] for even-hole-free graphs, and from [8] for claw-free graphs, one can easily prove the following two theorems. Similar to Definition 3, for an independent set A of a graph G with $|A| \geq k$, we denote $\mu_k^A(G) = \max\{|J| : A \leftrightarrow_k^G J\}$.

Theorem 15. *(*) Let A and B be independent sets of an even-hole-free or claw-free graph G. Then in polynomial time, it can be decided whether $A \leftrightarrow_k^G B$.*

Theorem 16. *(*) Let A be an independent set of a graph G that is even-hole-free or claw-free. Then $\mu_k^A(G) = |A|$ if A is a dominating set of size k, and $\mu_k^A(G) = \alpha(G)$ otherwise.*

It follows that for an even-hole-free or claw-free graph G, $\mu_k^A(G)$ can be computed efficiently if $\alpha(G)$ can be computed efficiently. Unfortunately, for even-hole-free graphs G it is an open question whether this can be done (see [22,26]). Nevertheless, for the subclass of chordal graphs, an efficient algorithm to compute $\alpha(G)$ is known [15]. For claw-free graphs, $\alpha(G)$ can be computed efficiently as well [23,25]. Denote by $\mathcal{G}^*$ class of all graphs that are chordal or claw-free. We conclude that if for G, a cotree decomposition into $\mathcal{G}^*$-graphs is given, then the conditions of Theorem 4 are satisfied, and thus the TAR-Reachability problem can be solved efficiently for G. It only remains to find such a cotree decomposition efficiently. Recall that for a graph G, by $\overline{G}$ the *complement* of G is denoted, which is the graph $\overline{G} = (V(G), \{uv \mid uv \notin E(G)\})$.

Definition 17. *A* maximal cotree decomposition *is a generalized cotree decomposition T where for every leaf $u \in V(T)$, both G_u and $\overline{G_u}$ are connected.*

Proposition 18. *(*) For any graph G, a maximal cotree decomposition of G can be computed in polynomial time.*

A graph class $\mathcal{G}$ is called *hereditary* if for every $G \in \mathcal{G}$ and every induced subgraph H of G, $H \in \mathcal{G}$ holds. Clearly, the aforementioned class $\mathcal{G}^*$ is hereditary.

Lemma 19. *(*) Let $\mathcal{G}$ be a hereditary graph class, and let G be a graph that admits a cotree decomposition into $\mathcal{G}$-graphs. Then every maximal cotree decomposition of G is a cotree decomposition into $\mathcal{G}$-graphs.*

From Proposition 18 and Lemma 19 it follows that a cotree decomposition into $\mathcal{G}^*$-graphs can be computed in polynomial time. Together, these statements yield the main result of this section.

Theorem 20. *(*) Let G be a graph that admits a cotree decomposition into graphs that are chordal or claw-free, and let A and B be independent sets of G, both of size at least k. Then in polynomial time, we can decide whether $A \leftrightarrow_k^G B$.*

7 Discussion

In the full version of this paper [6], we show that our DP algorithm for cographs G can be implemented to run in time $O(n^2)$, where $n = |V(G)|$. The key to this is a more efficient computation of stable tuples, not based on Definition 9. Secondly, in [6] we show that components of $\mathrm{TAR}_k(G)$ have diameter at most $4n - 2k$, if G is a cograph.

The following question related to independent set reconfiguration in cographs is still open: what is the complexity of deciding whether there exists a k-TAR-sequence of length at most ℓ between two independent sets of a cograph? (Recall that for general graphs, this is strongly NP-hard [22].) In [6], we also asked if it can be decided efficiently whether $\mathrm{TAR}_k(G)$ is connected, using similar techniques. In subsequent research [2], this question has been answered affirmatively.

References

1. Bonamy, M., Bousquet, N.: Recoloring bounded treewidth graphs. Electron. Notes Discrete Math. **44**, 257–262 (2013)
2. Bonamy, M., Bousquet, N.: Reconfiguring independent sets in cographs, June 2014. arXiv:1406.1433
3. Bonamy, M., Johnson, M., Lignos, I., Patel, V., Paulusma, D.: Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. J. Comb. Optim. 1–12 (2012)
4. Bonsma, P.: Rerouting shortest paths in planar graphs. In: FSTTCS 2012. vol. 18, LIPIcs, pp. 337–349. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
5. Bonsma, P.: The complexity of rerouting shortest paths. Theor. Comput. Sci. **510**, 1–12 (2013)
6. Bonsma, P.: Independent set reconfiguration in cographs, February 2014. arXiv:1402.1587
7. Bonsma, P., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. Theor. Comput. Sci. **410**(50), 5215–5226 (2009)
8. Bonsma, P., Kamiński, M., Wrochna, M.: Reconfiguring independent sets in claw-free graphs. In: Ravi, R., Gørtz, I.L. (eds.) SWAT 2014. LNCS, vol. 8503, pp. 86–97. Springer, Heidelberg (2014)
9. Cereceda, L., van den Heuvel, J., Johnson, M.: Connectedness of the graph of vertex-colourings. Discrete Appl. Mathe. **308**(5–6), 913–919 (2008)
10. Cereceda, L., van den Heuvel, J., Johnson, M.: Mixing 3-colourings in bipartite graphs. Eur. J. Comb. **30**(7), 1593–1606 (2009)

11. Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colorings. J. Graph Theory **67**(1), 69–82 (2011)
12. Corneil, D., Perl, Y., Stewart, L.: A linear recognition algorithm for cographs. SIAM J. Comput. **14**(4), 926–934 (1985)
13. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Appl. Math. **101**(13), 77–114 (2000)
14. Eggermont, C., Woeginger, G.J.: Motion planning with pulley, rope, and baskets. Theory Comput. Syst. **53**(4), 569–582 (2013)
15. Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. **1**(2), 180–187 (1972)
16. Gopalan, P., Kolaitis, P.G., Maneva, E., Papadimitriou, C.H.: The connectivity of boolean satisfiability: computational and structural dichotomies. SIAM J. Comput. **38**(6), 1863–1920 (2009)
17. Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theor. Comput. Sci. **343**(1–2), 72–96 (2005)
18. van den Heuvel, J.: The complexity of change. Surv. Comb. **2013**, 127–160 (2013)
19. Ito, T., Demaine, E.D.: Approximability of the subset sum reconfiguration problem. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 58–69. Springer, Heidelberg (2011)
20. Ito, T., Demaine, E.D., Harvey, N.J.A., Papadimitriou, C.H., Sideri, M., Uehara, R., Uno, Y.: On the complexity of reconfiguration problems. Theor. Comput. Sci. **412**(12–14), 1054–1065 (2011)
21. Kamiński, M., Medvedev, P., Milanič, M.: Shortest paths between shortest paths. Theor. Comput. Sci. **412**(39), 5205–5210 (2011)
22. Kamiński, M., Medvedev, P., Milanič, M.: Complexity of independent set reconfigurability problems. Theor. Comput. Sci. **439**, 9–15 (2012)
23. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory Ser. B **28**(3), 284–304 (1980)
24. Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki, A.: On the parameterized complexity of reconfiguration problems. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 281–294. Springer, Heidelberg (2013)
25. Sbihi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discrete Math. **29**(1), 53–76 (1980)
26. Vušković, K.: Even-hole-free graphs: a survey. Appl. Anal. Discrete Math. **4**(2), 219–240 (2010)
27. Wrochna, M.: Reconfiguration in bounded bandwidth and treedepth, May 2014. arXiv:1405.0847