Les Cahiers du GERAD ISSN: 0711–2440

Automated Generation of Conjectures on Forbidden Subgraph Characterization

C. Desrosiers, Ph. Galinier,
P. Hansen, A. Hertz

G–2007–48

July 2007

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds québécois de la recherche sur la nature et les technologies.

Automated Generation of Conjectures on Forbidden Subgraph Characterization

Christian Desrosiers
GERAD and École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
christian.desrosiers@polymtl.ca

Philippe Galinier
CRT and École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
philippe.galinier@polymtl.ca

Pierre Hansen
GERAD and HEC Montréal
Montréal (Québec) Canada, H3T 2A7
pierre.hansen@gerad.ca

Alain Hertz
GERAD and École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
alain.hertz@gerad.ca

July 2007

Les Cahiers du GERAD

G–2007–48

Copyright © 2007 GERAD

Abstract

Given a class of graphs $\mathcal{F}$, a forbidden subgraph characterization (FSC) is a set of graphs $\mathcal{H}$ such that a graph G belongs to $\mathcal{F}$ if and only if no graph of $\mathcal{H}$ is isomorphic to an induced subgraph of G. FSCs play a key role in graph theory, and are at the center of many important results obtained in that field. In this paper, we present novel methods that automate the generation of conjectures on FSCs, or conditions to have an FSC. We also use these methods to reproduce some known results of graph theory, as well as to discover new ones.

Key Words: Graph theory, forbidden subgraph characterization, automated conjecture generation.

Résumé

Étant donné une classe de graphes $\mathcal{F}$, une caractérisation par sous-graphes interdits (CSI) est un ensemble de graphes $\mathcal{H}$ tel qu'un graphe G appartient à $\mathcal{F}$ si et seulement si aucun graphe de $\mathcal{H}$ est isomorphe à un sous-graphe induit de G. Les CSI jouent un rôle clé en théorie des graphes et sont au cœur de plusieurs résultats importants de ce domaine. Dans cet article, nous présentons de nouvelles méthodes qui automatisent la génération de conjectures portant sur les CSI ou sur des conditions pour avoir un CSI. Nous utilisons également ces méthodes pour reproduire des résultats connus de la théorie des graphes ainsi qu'en découvrir de nouveaux.

1 Introduction

Traditionally, the process of discovering new knowledge in graph theory was carried out by mathematicians, with little assistance from computers. Yet, in recent years, mathematicians in that field have turned to computers to find some very important results. A famous illustration of this is the proof to the four color conjecture, which was done in large part by computers [16]. Since then, computers have played an increasing role in the discovery of new knowledge in graph theory, and many tools for this task have been proposed. One of the first computer programs for this purpose is *Graffiti*, developed by Fajtlowicz [6], which has generated over a thousand conjectures as algebraic equations involving graph invariants. Another more recent and equally prolific tool to generate conjectures involving graph invariants is *AGX*, proposed by Caporossi and Hansen [4]. This last program, which applies the Variable Neighborhood Search metaheuristic [14] to find extremal graphs, can also be used to find graph satisfying various constraints, to find structural conjectures, to refute conjectures and to suggest proofs.

Although automating the generation of conjectures has been the aim of many works, almost all these focused on generating conjectures in the form of relations on graph invariants. Yet, as recently suggested by Hansen et al. in [11], there are many interesting results in graph theory that take a different form. One of them, known as *forbidden subgraph characterization* (FSC), describes a class of graphs in terms of the subgraphs that these graphs are not allowed to have. A well known FSC, due to Seymour et al. [5], characterizes perfect graphs as the graphs which do not have as induced subgraph any odd cycle containing five or more vertices, or its complement. Another important FSC, due to Beineke [2], characterizes line graphs using nine forbidden graphs, shown in Figure 1.

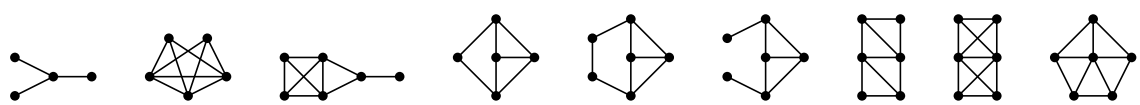


Figure 1: A forbidden subgraph characterization of line graphs.

In this paper, we present some new methods to automatically generate conjectures on FSCs. The rest of the paper is structured as follows. We first introduce some preliminary concepts that will help to understand the rest of the paper. We then describe our methods, by considering three problems: finding sufficient conditions for an FSC, finding necessary conditions for an FSC, and finding actual FSCs. We then show how these methods can be used in practice to generate conjectures, and illustrate this by reproducing some known results, as well as generating new ones. Finally, we end this paper with a short summary of our work.

2 Preliminary concepts and definitions

Let $\mathcal{G}$ be the set containing all graphs. A class of graphs $\mathcal{F} \subseteq \mathcal{G}$ is a possibly infinite set of graphs that share a common property. Let $\mathcal{H}$ be a set of graphs, we say that a graph G is $\mathcal{H}$-free if there is no graph of $\mathcal{H}$ isomorphic to one of its induced subgraphs, and write $\mathcal{G}_{\mathcal{H}}$ the set of all such graphs. Using this terminology, an FSC of $\mathcal{F}$ is a set of graphs $\mathcal{H}$ such that $\mathcal{G}_{\mathcal{H}} = \mathcal{F}$. As we will see, not every class of graphs has FSCs. For classes that do not have an FSC, we are often interested in finding some weaker rules allowing to partially characterize the graphs of these classes. These rules come in two forms: *sufficient conditions* (SFSC) and *necessary* conditions (NFSC). Let $\mathcal{F}$ be the class of graphs to characterize and $\mathcal{H}$ be a set of forbidden subgraphs. Sufficient conditions can be expressed as follows: if a graph G is $\mathcal{H}$-free, then it is part of $\mathcal{F}$. Thus, a sufficient condition is such that $\mathcal{G}_{\mathcal{H}} \subseteq \mathcal{F}$. However, sufficient conditions do not fully describe $\mathcal{F}$. Indeed, if G is not $\mathcal{H}$-free, we cannot use this type of condition to determine if G is in $\mathcal{F}$ or not. On the other hand, necessary conditions can be expressed as follows: if a graph G is in $\mathcal{F}$ then it is $\mathcal{H}$-free. This implies that $\mathcal{G}_{\mathcal{H}} \supseteq \mathcal{F}$. Again, necessary conditions offer a partial description of $\mathcal{F}$: if G is not in $\mathcal{F}$ then it can either be $\mathcal{H}$-free or not.

Let G be a graph, and $W = \{v_1, v_2, \ldots, v_q\}$ be a subset of $V(G)$. We write $G[W]$ or, when the context is clear, $\langle v_1, v_2, \ldots, v_q \rangle$ the subgraph of G induced by W. Furthermore, let H be another graph, we write $G \simeq H$ when G is isomorphic to H, and $H \subseteq G$ when H is isomorphic to an induced subgraph of G. The following elementary properties will be used later on to prove more complex results.

Property 1 *Let G_1, G_2, G_3 be three graphs, and $\mathcal{H}_1$, $\mathcal{H}_2$ be two sets of forbidden subgraphs.*

(a). If $G_1 \subseteq G_2$ and $G_2 \subseteq G_3$ then $G_1 \subseteq G_3$.
(b). If $G_1 \nsubseteq G_2$ and $G_3 \subseteq G_2$ then $G_1 \nsubseteq G_3$.
(c). If $G_1 \nsubseteq G_2$ and $G_1 \subseteq G_3$ then $G_3 \nsubseteq G_2$.
(d). If $G_1 \subseteq G_2$ then $\mathcal{G}_{\{G_1\}} \subseteq \mathcal{G}_{\{G_2\}}$.
(e). $\mathcal{G}_{\mathcal{H}_1 \cup \mathcal{H}_2} = \mathcal{G}_{\mathcal{H}_1} \cap \mathcal{G}_{\mathcal{H}_2}$.
(f). If $\mathcal{H}_1 \subseteq \mathcal{H}_2$ then $\mathcal{G}_{\mathcal{H}_1} \supseteq \mathcal{G}_{\mathcal{H}_2}$.
(g). If $G_1 \subseteq G_2$ then $\mathcal{G}_{\{G_1\}} = \mathcal{G}_{\{G_1, G_2\}}$.

To lighten the presentation, we will, in the rest of this paper, refer to induced subgraphs simply as subgraphs.

3 Sufficient conditions

Formally, we write an SFSC as G is $\mathcal{H}$-free $\Rightarrow G \in \mathcal{F}$. This expression is logically equivalent to $G \in \overline{\mathcal{F}} \Rightarrow \exists H \in \mathcal{H}$ s.t. $H \subseteq G$, where $\overline{\mathcal{F}} = \mathcal{G} \setminus \mathcal{F}$ is the complement of the graph class

$\mathcal{F}$. The task of finding an SFSC can thus be defined as follows: find a set of graphs $\mathcal{H}$ such that

$$\forall G \in \overline{\mathcal{F}},\ \exists H \in \mathcal{H} \text{ s.t. } H \subseteq G.$$

While a graph class $\mathcal{F}$ can have many SFSCs, these may not be equally useful. For instance, $\mathcal{H} = \overline{\mathcal{F}}$ is an SFSC of $\mathcal{F}$, but is as complex as the class $\mathcal{F}$ itself. Moreover, let H be the graph composed of a single vertex, $\mathcal{H} = \{H\}$ is an SFSC of $\mathcal{F}$ since H is a subgraph of all graphs of $\overline{\mathcal{F}}$. However, $\mathcal{H}$ offers no real information on $\mathcal{F}$, since each graph of $\mathcal{F}$ also has H as subgraph, i.e. $\mathcal{F} \cap \mathcal{G}_{\mathcal{H}} = \emptyset$. To help us find useful SFSCs, we need to introduce two measures. The *complexness* of an SFSC roughly evaluates the amount of information needed by this SFSC to describe $\mathcal{F}$. Let $\mathcal{H}$, $\mathcal{H}'$ be two SFSCs of $\mathcal{F}$. We say that $\mathcal{H}$ is more complex than $\mathcal{H}'$ if $|\mathcal{H}| > |\mathcal{H}'|$, or in the case where $\mathcal{H} = \{H\}$ and $\mathcal{H}' = \{H'\}$, if $|V(H)| > |V(H')|$. On the other hand, the tightness of an SFSC evaluates how well it describes $\mathcal{F}$. We say that an SFSC $\mathcal{H}$ is tighter than another SFSC $\mathcal{H}'$ if $\mathcal{G}_{\mathcal{H}'} \subseteq \mathcal{G}_{\mathcal{H}}$. Thus, an SFSC $\mathcal{H}$ is maximally tight if $\mathcal{G}_{\mathcal{H}} = \mathcal{F}$. We see that the concepts of complexness and tightness are related: a tighter SFSC can generally be obtained by increasing its complexness. Hence, the measure of an SFSC's usefulness is a compromise between the minimization of its complexness and the maximization of its tightness.

From the above definitions, we can impose an additional constraint on the selection of an SFSC $\mathcal{H}$. Suppose $\mathcal{H}$ contains two graphs H and H' such that $H' \subset H$. From Property 1.(g), we know that $\mathcal{H}' = \mathcal{H} \setminus \{H\}$ is an SFSC of $\mathcal{F}$ as tight as $\mathcal{H}$. However, $\mathcal{H}'$ is less complex than $\mathcal{H}$ since it contains one less graph. We can thus limit our search to the sets of graphs $\mathcal{H}$ such that

$$\forall H \in \mathcal{H},\ \nexists H' \in \mathcal{H} \text{ s.t. } H' \subset H.$$

In the rest of the paper, we will call *minimal* a set $\mathcal{H}$ that satisfies the above property, and denote $\min(\mathcal{H}) \subseteq \mathcal{H}$ the subset of $\mathcal{H}$ that is minimal.

Proposition 1 *Let $\mathcal{F}$ be a class of graphs and $\mathcal{H}$ be a minimal SFSC of $\mathcal{F}$ with maximum tightness, then $\mathcal{H} = \min(\overline{\mathcal{F}})$.*

Proof. Let $\mathcal{H} = \min(\overline{\mathcal{F}})$. We first show that $\mathcal{H}$ is an SFSC of $\mathcal{F}$. Suppose it is not, then there is a graph $G \in \overline{\mathcal{F}} \setminus \mathcal{H}$ such that $\nexists H \in \mathcal{H}$, $H \subset G$. However, G would then be, by definition, in $\mathcal{H}$, which is a contradiction. We next show that $\mathcal{H}$ is the only SFSC with maximum tightness. Let $\mathcal{H}'$ be any minimal SFSC of $\mathcal{F}$. For each $H' \in \mathcal{H}'$, if $\nexists G \in \overline{\mathcal{F}}$ such that $H' \subseteq G$ then, clearly, the tightness of $\mathcal{H}'$ is not maximum since $\mathcal{H}' \setminus \{H'\}$ is an SFSC of greater tightness. Otherwise, if $H' \in \mathcal{F}$, let $\mathcal{B}$ denote the graphs of $\overline{\mathcal{F}}$ which have H' as subgraph. By definition, $\mathcal{H}'' = \mathcal{H}' \setminus \{H'\} \cup \mathcal{B}$ is an SFSC of $\mathcal{F}$. Furthermore, from Properties 1.(d) and 1.(e), we have that $\mathcal{G}_{\{H'\}} \subseteq \mathcal{G}_{\mathcal{B}}$ and that $\mathcal{G}_{\mathcal{H}'} \subseteq \mathcal{G}_{\mathcal{H}''}$. Finally, since $H' \notin \mathcal{G}_{\mathcal{H}'}$ and $H' \in \mathcal{G}_{\mathcal{H}''}$, we conclude that $\mathcal{G}_{\mathcal{H}'} \subset \mathcal{G}_{\mathcal{H}''}$ and therefore that $\mathcal{H}''$ is tighter than $\mathcal{H}'$. The remaining case is $H' \in \overline{\mathcal{F}}$. Suppose that $H' \notin \mathcal{H}$, then by definition there is a graph $H \in \mathcal{H}$ such that $H \subset H'$. Furthermore, since $\mathcal{H}'$ is an SFSC of $\mathcal{F}$, there must be a graph $H'' \in \mathcal{H}'$ such that $H'' \subseteq H \subset H'$, which contradicts the minimality of $\mathcal{H}'$.

Therefore, we have that $H' \in \mathcal{H}$, and thus, either $\mathcal{H}'$ does not have maximum tightness or $\mathcal{H}' = \mathcal{H}$. $\square$

3.1 Single graph SFSC

Before we tackle the general task of finding an SFSC having an arbitrary number of graphs, we first consider the simpler case of finding an SFSC of a graph class $\mathcal{F}$, containing a single graph. In this case, the problem can be formulated as finding a set $\mathcal{H} = \{H\}$ such that $\forall G \in \overline{\mathcal{F}},\ H \subseteq G$. Thus, H is a common subgraph of the graphs in $\overline{\mathcal{F}}$. Furthermore, let H, H' be two graphs such that $H \subset H'$. From Property 1.(d), we have $\mathcal{G}_{\{H\}} \subset \mathcal{G}_{\{H'\}}$ and therefore that $\{H'\}$ is tighter than $\{H\}$. Hence, if we want to maximize the tightness, we should find a common subgraph H which is maximal w.r.t. inclusion. This principle serves as the main idea of our first algorithm. This algorithm searches for a graph H_k which is contained in all graphs of $\overline{\mathcal{F}}$, in the following way. Starting with a graph H_0 chosen in $\overline{\mathcal{F}}$, the algorithm searches, at every iteration k, for a graph $G_k \in \overline{\mathcal{F}}$ which does not contain H_k. If such a graph exists, the algorithm finds a new graph H_{k+1}, which is a subgraph of G_k as well as of $H_0, G_0, G_1, \ldots, G_{k-1}$. Otherwise, if no such graph exists, we know that H_k is a subgraph of all graphs of $\overline{\mathcal{F}}$, and thus is an SFSC of $\mathcal{F}$. Let $\mathcal{L}$ be a finite set of graphs and maxCS($\mathcal{L}$) denote the set of common subgraphs of $\mathcal{L}$ which have a maximum number of vertices. The algorithm to find a single graph SFSC is formally expressed in the following procedure:

1. Choose any H_0 in $\overline{\mathcal{F}}$;
2. Let $\mathcal{L}_0 := \{H_0\}$, and $k := 0$;
3. If $\exists G_k \in \overline{\mathcal{F}} \cap \mathcal{G}_{\{H_k\}}$ then
   3.1. Let $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{G_k\}$;
   3.2. Choose H_{k+1} in maxCS($\mathcal{L}_{k+1}$);
   3.3. Let $k := k + 1$ and goto (*3*);
4. Else, return $\mathcal{H} = \{H_k\}$;

Proposition 2 *The above algorithm produces, in a finite number of steps, a common subgraph of $\overline{\mathcal{F}}$ which is maximal w.r.t. inclusion.*

Proof. We first prove that the algorithm terminates in a finite number of steps. Consider any two graphs H_i, H_j such that $i < j$. Since H_j is a common subgraph of $\mathcal{L}_j$, and $G_i \in \mathcal{L}_j$, we have $H_j \subseteq G_i$ (*1*). Moreover, since $G_i \in \overline{\mathcal{F}} \cap \mathcal{G}_{\{H_i\}}$, we know that $H_i \not\subseteq G_i$ (*2*). Combining (*1*) and (*2*), we get $H_i \neq H_j$. Furthermore, since H_i and H_j are maximum common subgraphs of $\mathcal{L}_i$ and $\mathcal{L}_j$, and since $\mathcal{L}_i \subset \mathcal{L}_j$, we have that $|V(H_j)| \leq |V(H_i)|$. Finally, since there is a finite number of graphs of $|V(H_i)|$ or less vertices, the algorithm will terminate in a finite number of steps. Next, we prove that the graph H_k returned by the algorithm is a common subgraph of $\overline{\mathcal{F}}$. Suppose it is not. Then, there exists a graph

$G_k \in \overline{\mathcal{F}}$ that does not have H_k as subgraph. However, this is a contradiction since the algorithm terminates only when no such graph exists. Finally, we show that H_k is maximal w.r.t. inclusion. Suppose this is not the case. Then, there exists a common subgraph H' of $\overline{\mathcal{F}}$, such that $H_k \subset H'$. Since H' is a common subgraph of $\overline{\mathcal{F}}$, it is also a common subgraph of $\mathcal{L}_k \subseteq \overline{\mathcal{F}}$. However, this contradicts the fact that H_k is a maximum common subgraph of $\mathcal{L}_k$. $\square$

3.2 Multiple graph SFSC

From Proposition 1, we know that if a graph H, obtained by the single graph SFSC algorithm, is not in $\overline{\mathcal{F}}$, then we can find a tighter SFSC containing more than one forbidden subgraph. Let M be the maximum number of graphs we want to have in the SFSC. If $|\min(\overline{\mathcal{F}})| \leq M$, the algorithm should return $\min(\overline{\mathcal{F}})$ since this SFSC has maximum tightness. Otherwise, we search for an SFSC of $\mathcal{F}$ in the following way. We first find a set $\mathcal{M}$ containing $M-1$ of the smallest graphs of $\overline{\mathcal{F}}$. Since $\mathcal{M} = \min(\mathcal{M}) \subset \min(\overline{\mathcal{F}})$, by definition, $\mathcal{M}$ is an SFSC of $\mathcal{G}_{\mathcal{M}} \supset \mathcal{F}$ that has maximum tightness. We then find a common subgraph H of $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M}}$. Finally, we remove from $\mathcal{M}$ the graphs that have H as subgraph and return the SFSC $\mathcal{H} = \mathcal{M} \cup \{H\}$. Let $\mathcal{C}$ be a given graph class and L an integer, the following procedure returns the set $\min(\mathcal{C})$, if $|\min(\mathcal{C})| \leq L$, or returns L of the smallest graphs of $\min(\mathcal{C})$:

1. Let $\mathcal{M}_0 := \emptyset$ and $l := 0$;
2. If $\mathcal{C} \cap \mathcal{G}_{\mathcal{M}_l} \neq \emptyset$ and $l < L$ then
   - 2.1. Choose $H_l \in \mathcal{C} \cap \mathcal{G}_{\mathcal{M}_l}$ with minimum order;
   - 2.2. Let $\mathcal{M}_{l+1} := \mathcal{M}_l \cup \{H_l\}$;
   - 2.3. Let $l := l+1$ and goto (*2*);
3. Else, return $\mathcal{M} = \mathcal{M}_l$;

Denote by $\text{findMin}(\mathcal{C}, L)$ the set of graphs returned by the above procedure, for a class $\mathcal{C}$ and a maximum number of graphs L. The following procedure finds an SFSC for a graph class $\mathcal{F}$, containing at most L graphs:

1. Let $\mathcal{M} := \text{findMin}(\overline{\mathcal{F}}, M-1)$ and choose any H_0 in $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M}}$;
2. Let $\mathcal{L}_0 := \{H_0\}$, and $k := 0$;
3. If $\exists G_k \in \overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M} \cup \{H_k\}}$ then
   - 3.1. Let $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{G_k\}$;
   - 3.2. Choose H_{k+1} in $\text{maxCS}(\mathcal{L}_{k+1})$;
   - 3.3. Let $k := k+1$ and goto (*3*);
4. Else, return $\mathcal{H} = \min(\mathcal{M} \cup \{H_k\})$;

Proposition 3 *The above algorithm produces, in a finite number of steps, an SFSC of $\mathcal{F}$ containing at most M graphs.*

Proof. We first show that the algorithm terminates in a finite number of steps. Clearly, the procedure that returns findMin($\overline{\mathcal{F}}, M-1$) will terminate in at most $n-1$ iterations. Furthermore, from Proposition 2, we know that finding the last graph can be done in a finite number of steps. Thus, the algorithm is finite. We next prove that the set $\mathcal{H} = \min(\mathcal{M} \cup \{H_k\})$ returned by the algorithm is an SFSC of at most M graphs. Consider once again the procedure findMin. Since we have $|\mathcal{M}_l| = l$, at every iteration l, this procedure will return a set $\mathcal{M}$ containing at most $M-1$ graphs. Furthermore, suppose that $\mathcal{M} \cup \{H_k\}$ is not an SFSC. Then, there exists a graph $G \in \overline{\mathcal{F}}$ that does not have H_k nor any graph of $\mathcal{M}_k$ as subgraph. However, this is a contradiction since the algorithm terminates only when no such graph exists. Thus, $\mathcal{M} \cup \{H_k\}$ is an SFSC of $|\mathcal{M}| + 1 \leq M$ graphs. Finally, since $\min(\mathcal{M} \cup \{H_k\}) \subseteq \mathcal{M} \cup \{H_k\}$, we have, by definition, that $\mathcal{H}$ is an SFSC of at most M graphs. □

Proposition 4 *Let H be a forbidden subgraph obtained by the single graph SFSC algorithm, and $\mathcal{H} = \mathcal{M} \cup \{H_k\}$ be the multiple graph SFSC returned by the above algorithm. If $H \subset H_k$ then $\mathcal{H}$ is a tighter SFSC than $\{H\}$.*

Proof. Since $\{H\}$ is an SFSC and $\mathcal{M} \subseteq \overline{\mathcal{F}}$, we know that $H \subseteq H'$, for all $H' \in \mathcal{M}$, and, from Properties 1.(d) and 1.(e), that $\mathcal{G}_{\{H\}} \subseteq \mathcal{G}_{\mathcal{M}}$. Furthermore, since $H \subset H_k$, we have $\mathcal{G}_{\{H\}} \subset \mathcal{G}_{\{H_k\}}$, and thus that $\mathcal{G}_{\{H\}} \subset \mathcal{G}_{\mathcal{M}} \cap \mathcal{G}_{\{H_k\}} = \mathcal{G}_{\mathcal{H}}$. Therefore $\mathcal{H}$ is a tighter SFSC than $\{H\}$. □

4 Necessary conditions

Let $\mathcal{F}$ be a graph class, an NFSC is a set of graphs $\mathcal{H}$ such that all graphs in $\mathcal{F}$ are $\mathcal{H}$-free. Finding an NFSC thus amounts to finding $\mathcal{H}$ such that

$$\forall G \in \mathcal{F}, \ \nexists H \in \mathcal{H} \text{ s.t. } H \subseteq G.$$

This implies that $\mathcal{H} \subseteq \overline{\mathcal{F}}$. As we did for SFSCs, we can use the criteria of complexness and tightness to guide our search of an NFSC. However, since an NFSC $\mathcal{H}$ is such that $\mathcal{G}_{\mathcal{H}} \supseteq \mathcal{F}$, the definition of tightness differs from the one given for sufficient conditions. Let $\mathcal{H}$, $\mathcal{H}'$ be two NFSCs, we say that $\mathcal{H}$ is tighter than $\mathcal{H}'$ if $\mathcal{G}_{\mathcal{H}} \subset \mathcal{G}_{\mathcal{H}'}$. As we have done for sufficient conditions, we first present a method that finds single graph NFSCs and then generalize this method to multiple graph NFSCs.

4.1 Single graph NFSC

The task of finding a single graph NFSC of a graph class $\mathcal{F}$ corresponds to finding a graph H such that, for all $G \in \mathcal{F}$, $H \not\subseteq G$. Thus, H is a common "non-subgraph" of the graphs

in $\mathcal{F}$. Moreover, let H and H' be two graphs such that $H \subset H'$. From Property 1.(d), we have that $\mathcal{G}_{\{H\}} \subset \mathcal{G}_{\{H'\}}$ and, therefore, that $\{H\}$ is tighter than $\{H'\}$. Thus, to maximize the tightness, H should be a common non-subgraph which is minimal w.r.t. inclusion. Our single graph NFSC algorithm is based on this idea. Let $\overline{\mathcal{G}}_{\mathcal{H}} = \mathcal{G} \setminus \mathcal{G}_{\mathcal{H}}$ be the set of graphs containing at least one graph of $\mathcal{H}$. Starting with a graph H_0 that has a single vertex, at each iteration k, we try to find a graph $G_k \in \mathcal{F}$ which contains H_k. If such a graph exists, then H_k is not an NFSC, and we improve H_k with a graph H_{k+1} that is not a subgraph of any of the $G_0, \ldots, G_{k-1}$ nor of G_k. We repeat this process until no graph of $\mathcal{F}$ contains H_k, i.e. when $\mathcal{F} \cap \overline{\mathcal{G}}_{\{H_k\}} = \emptyset$.

In the case where $\mathcal{F}$ has no NFSC, we may always be able to find $G_k \in \mathcal{F}$ containing H_k, and thus, the algorithm may never stop. To avoid this problem, we impose a limit N on the number of vertices of our NFSC. Let $\mathcal{L}$ be a set of graphs, we denote minCNS($\mathcal{L}$) the set of common non-subgraphs of $\mathcal{L}$ which have a minimum number of vertices. The single graph NFSC algorithm can be summarized as follows:

1. Let H_0 be a single vertex graph, $\mathcal{L}_0 := \emptyset$, and $k := 0$;
2. If $\exists G_k \in \mathcal{F} \cap \overline{\mathcal{G}}_{\{H_k\}}$ and $|V(H_k)| \leq N$
   2.1. Let $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{G_k\}$;
   2.2. Choose H_{k+1} in minCNS($\mathcal{L}_{k+1}$);
   2.3. Let $k := k + 1$ and goto (*2*);
3. Else, return $\mathcal{H} = \{H_k\}$;

Proposition 5 *The above algorithm produces, in a finite number of steps, a common non-subgraph of $\mathcal{F}$ which is minimal w.r.t. inclusion and has at most N vertices, if one exists. Otherwise, the algorithm returns a graph that has more than N vertices.*

Proof. We first prove that the algorithm terminates in a finite number of steps. Consider any two graphs H_i and H_j such that $i < j$. Since H_j is a common non-subgraph of $\mathcal{L}_j$, and $G_i \in \mathcal{L}_j$, we have $H_j \not\subseteq G_i$ (*1*). Moreover, since $G_i \in \mathcal{F} \cap \overline{\mathcal{G}}_{\{H_i\}}$, we know that $H_i \subseteq G_i$ (*2*). Combining (*1*) and (*2*), we get $H_i \neq H_j$. Furthermore, since H_i and H_j are minimum common non-subgraphs of $\mathcal{L}_i$ and $\mathcal{L}_j$, and since $\mathcal{L}_i \subset \mathcal{L}_j$, we have that $|V(H_j)| \geq |V(H_i)|$. Finally, since there is a finite number of graphs of N or less vertices, the algorithm will terminate in a finite number of steps. Next, we prove that, if the graph H_k returned by the algorithm is such that $|V(H_k)| \leq N$, then it is a common non-subgraph of $\mathcal{F}$. Suppose it is not. Then, there exists a graph $G_k \in \mathcal{F}$ that has H_k as subgraph. However, this is a contradiction since the algorithm terminates only when no such graph exists, or if $|V(H_k)| > N$. Finally, we show that H_k is minimal w.r.t. inclusion. Suppose this is not the case. Then, there exists a common non-subgraph H' of $\mathcal{F}$ such that $H' \subset H_k$. Since H' is a common non-subgraph of $\mathcal{F}$, it is also a common non-subgraph of $\mathcal{L}_k \subseteq \mathcal{F}$. However, this contradicts the fact that H_k is a minimum common non-subgraph of $\mathcal{L}_k$. $\square$

Although the algorithm guarantees to find in a finite number of steps a single graph NFSC, if one exists, we can accelerate its convergence using the fact that all NFSCs $\mathcal{H}$ are such that $\mathcal{H} \subseteq \overline{\mathcal{F}}$. Thus, instead of starting with a graph H_0 composed of a single vertex, we can start with H_0 as the smallest graph of $\overline{\mathcal{F}}$. Furthermore, let $\text{minCNS}(\mathcal{L}, \mathcal{C})$ be the smallest graph of a class $\mathcal{C}$ that is not contained in any graph of $\mathcal{L}$, we can choose H_{k+1} in $\text{minCNS}(\mathcal{L}, \overline{\mathcal{F}})$ instead of $\text{minCNS}(\mathcal{L})$.

4.2 Multiple graph NFSC

As it was the case for sufficient conditions, we can sometimes improve the tightness of an NFSC by increasing its complexness.

Proposition 6 *Let $\mathcal{F}$ be a given graph class and let $\mathcal{N} = \{H \in \overline{\mathcal{F}} \mid \nexists G \in \mathcal{F} \text{ s.t. } H \subset G\}$. If $\mathcal{H}$ is a minimal NFSC of maximum tightness then $\mathcal{H} = \min(\mathcal{N})$.*

Proof. Let $\mathcal{H} = \min(\mathcal{N})$. We first show that $\mathcal{H}$ is an NFSC of $\mathcal{F}$. Suppose it is not, then there is a graph $G \in \mathcal{F} \setminus \mathcal{H}$ such that $\exists H \in \mathcal{H}$, $H \subset G$, which contradicts the definition of $\mathcal{H}$. We next show that $\mathcal{H}$ is the only NFSC with maximum tightness. Let $\mathcal{H}'$ be any minimal NFSC of $\mathcal{F}$, and let H' be any graph of $\mathcal{H}'$. By definition, we know that $H' \in \mathcal{N}$. Suppose that H' is not in $\mathcal{H}$, since $\mathcal{H} = \min(\mathcal{N})$, there is a graph $H \in \mathcal{H}$ such that $H \subset H'$. Furthermore, from Properties 1.(d) and 1.(e), we have that $\mathcal{G}_{\{H\}} \subset \mathcal{G}_{\{H'\}}$, and, thus, that the NFSC $\mathcal{H}'' = \mathcal{H}' \setminus \{H'\} \cup \{H\}$ is tighter than $\mathcal{H}'$. Therefore, $\mathcal{H}'$ is maximally tight only if $\mathcal{H}' \subseteq \mathcal{H}$. Suppose finally that $\mathcal{H}' \subset \mathcal{H}$. Then, by Property 1.($f$), we have that $\mathcal{G}_{\mathcal{H}'} \supset \mathcal{G}_{\mathcal{H}}$, and, thus, that $\mathcal{H}$ is tighter than $\mathcal{H}'$. Consequently, either $\mathcal{H}'$ does not have maximum tightness or $\mathcal{H}' = \mathcal{H}$. $\square$

Following the above proposition, the NFSC of a class $\mathcal{F}$ with maximum tightness $\mathcal{H}$ is simply the minimal set containing all individual graphs that are, by themselves, NFSCs. To find $\mathcal{H}$, we extend the previous algorithm such that it finds all the single graph NFSCs. We start with an empty set $\mathcal{H}$ and a graph H_0 having a single vertex, or with H_0 as the smallest graph of $\overline{\mathcal{F}}$ if we want to speed-up the converge. Then, at each iteration k, we look for a graph $G_k \in \mathcal{F}$ that contains H_k. If such a graph exists, we find the smallest graph H_{k+1} that is not contained in $G_0, \ldots, G_k$, nor contains any graph of $\mathcal{H}$. Again, if we wish to speed-up the convergence of the algorithm, we can restrict our search to $\overline{\mathcal{F}}$. However, if no such graph exists, H_k is a single graph NFSC and we add it to $\mathcal{H}$. We repeat this process until the order of H_k exceeds a given limit N. The multiple graph NFSC algorithm is summarized in the following pseudo-code:

1. Let H_0 be a single vertex graph;
2. Let $\mathcal{M}_0 := \emptyset$, $\mathcal{L}_0 := \emptyset$, and $k := 0$;
3. If $|V(H_k)| > N$ then return $\mathcal{H} = \mathcal{M}_k$;
4. Else, if $\exists G_k \in \mathcal{F} \cap \overline{\mathcal{G}}_{\{H_k\}}$ then let $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{G_k\}$ and $\mathcal{M}_{k+1} := \mathcal{M}_k$;

5. Else, let $\mathcal{M}_{k+1} := \mathcal{M}_k \cup \{H_k\}$ and $\mathcal{L}_{k+1} := \mathcal{L}_k$;
6. Choose H_{k+1} in $\text{minCNS}(\mathcal{L}_{k+1}, \mathcal{G}_{\mathcal{M}_{k+1}})$;
7. Let $k := k + 1$ and goto (*3*);

Proposition 7 *The above algorithm produces, in a finite number of steps, a set $\mathcal{H}$ containing all single graph NFSCs of at most N vertices.*

Proof. We first prove that the algorithm terminates in a finite number of steps. At every iteration k, only one of the conditions of steps (*3*), (*4*) and (*5*) is verified. If the condition of step (*4*) is verified, then, since $H_k \subseteq G_k$ and $H_{k+1} \nsubseteq G_k$, we have that $H_k \neq H_{k+1}$. Moreover, because H_k and H_{k+1} are minimum common non-subgraphs of $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$, and since $\mathcal{L}_k \subset \mathcal{L}_{k+1}$, we have that $|V(H_{k+1})| \geq |V(H_k)|$. However, if the condition of step (*5*) is verified, then we have $H_k \in \mathcal{M}_{k+1}$ and $H_{k+1} \in \mathcal{G}_{\mathcal{M}_{k+1}}$, and therefore $H_{k+1} \neq H_k$. Also, because H_k and H_{k+1} are minimum common non-subgraphs of $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$, and since $\mathcal{L}_k = \mathcal{L}_{k+1}$, we have $|V(H_{k+1})| \geq |V(H_k)|$. Finally, since there is a finite number of graphs with N or less vertices, the condition of step (*3*) will eventually be verified and the algorithm will terminate.

We next show that the algorithm is sound and complete. Since a graph H is only added to $\mathcal{H}$ if $\mathcal{F} \cap \overline{\mathcal{G}}_{\{H\}} = \emptyset$, we know that $\mathcal{F} \subseteq \mathcal{G}_{\{H\}}$ and thus that $\{H\}$ is an NFSC of $\mathcal{F}$. Furthermore, suppose there are two graphs $H_i, H_j \in \mathcal{H}$ such that $H_i \subset H_j$. Since, $|V(H_i)| < |V(H_j)|$, we know that $i < j$. Moreover, since $H_i \in \mathcal{M}_j$ and $H_j \in \mathcal{G}_{\mathcal{M}_j}$, we know that $H_i \nsubseteq H_j$ which is a contradiction. Therefore, the set $\mathcal{H}$ is minimal and the algorithm is sound. Moreover, suppose there exists a NFSC $\{H\}$ with $V(H) \leq N$, that is not found by the algorithm. By definition, we know that $H \in \text{minCNS}(\mathcal{F}, \mathcal{G}_{\mathcal{H}})$. Also, because minCNS finds the common non-subgraphs of minimum order, there must be an iteration k such that $H \notin \text{minCNS}(\mathcal{L}_k, \mathcal{G}_{\mathcal{M}_k})$. However, since $\mathcal{L}_k \subseteq \mathcal{F}$ and $\mathcal{M}_k \subseteq \mathcal{H}$, for every iteration k, we have $\text{minCNS}(\mathcal{F}, \mathcal{G}_{\mathcal{H}}) \subseteq \text{minCNS}(\mathcal{L}_k, \mathcal{G}_{\mathcal{M}_k})$, and thus $H \notin \text{minCNS}(\mathcal{F}, \mathcal{G}_{\mathcal{H}})$, which contradicts our initial supposition. Therefore, the algorithm is complete. □

5 Necessary and sufficient conditions

Before we present a method that finds FSCs, we need to answer some important questions. Firstly, we want to know under what conditions does a graph class have an FSC. The following proposition gives a well known result on the existence of an FSC for a given class of graphs, see e.g. [9].

Proposition 8 *A graph class $\mathcal{F}$ has an FSC if and only if it is hereditary.*

Proof. Let $\mathcal{H}$ be an FSC of $\mathcal{F}$. For any $G \in \mathcal{F}$, we know that G contains no graph of $\mathcal{H}$. Furthermore, for any $G' \subset G$, from Property 1.(*b*), we have that G' does not contain any graph of $\mathcal{H}$. Therefore, $G' \in \mathcal{G}_{\mathcal{H}} = \mathcal{F}$ and $\mathcal{F}$ is hereditary. Suppose conversely that $\mathcal{F}$ is an

hereditary class of graphs, and let G be a graph of $\mathcal{F}$. Since every subgraph of G belongs to $\mathcal{F}$, we know that no subgraph of G belongs to $\overline{\mathcal{F}}$, and thus that $G \in \mathcal{G}_{\overline{\mathcal{F}}}$. Therefore, we have that $\mathcal{F} \subseteq \mathcal{G}_{\overline{\mathcal{F}}}$ (*1*). Moreover, let H be any graph of $\mathcal{G}_{\overline{\mathcal{F}}}$. We know that $H \notin \overline{\mathcal{F}}$, and therefore that $H \in \mathcal{F}$. Consequently, we have that $\mathcal{G}_{\overline{\mathcal{F}}} \subseteq \mathcal{F}$ (*2*). Combining (*1*) and (*2*), we have $\mathcal{F} = \mathcal{G}_{\overline{\mathcal{F}}}$, and thus, $\overline{\mathcal{F}}$ is an FSC of $\mathcal{F}$. □

The next question is equally important: *does a graph class $\mathcal{F}$ have a unique FSC and, if so, what is it ?* This question is answered by the following propositions.

Proposition 9 *Let $\mathcal{H}$ and $\mathcal{H}'$ be two minimal forbidden subgraph characterizations, then $\mathcal{G}_{\mathcal{H}} = \mathcal{G}_{\mathcal{H}'}$ if and only if $\mathcal{H} = \mathcal{H}'$.*

Proof. Since $\mathcal{H} = \mathcal{H}' \Rightarrow \mathcal{G}_{\mathcal{H}} = \mathcal{G}_{\mathcal{H}'}$ is trivial, we only show that $\mathcal{G}_{\mathcal{H}} = \mathcal{G}_{\mathcal{H}'} \Rightarrow \mathcal{H} = \mathcal{H}'$. Suppose that $\mathcal{H} \neq \mathcal{H}'$, then there exists $H \in \mathcal{H}$ such that $H \notin \mathcal{H}'$, or $H' \in \mathcal{H}'$ such that $H' \notin \mathcal{H}$. Without loss of generality, suppose the first case in true. If there is no $H' \in \mathcal{H}'$ such that $H' \subset H$ then, for all $H'' \in \mathcal{H}'$, we know that $H \notin \mathcal{G}_{\{H\}}$ and that $H \in \mathcal{G}_{\{H''\}}$. Finally, from Property 1.(e), we have that $H \notin \mathcal{G}_{\mathcal{H}}$ and $H \in \mathcal{G}_{\mathcal{H}'}$, and therefore that $\mathcal{G}_{\mathcal{H}} \neq \mathcal{G}_{\mathcal{H}'}$. Otherwise, let H' be a graph in $\mathcal{H}'$ such that $H' \subset H$. There is no $H'' \in \mathcal{H}$ such that $H'' \subseteq H'$, otherwise we would have $H'' \subset H$ which contradicts the minimality of $\mathcal{H}$. Therefore, we have that $H' \notin \mathcal{G}_{\{H'\}}$ and that $H' \in \mathcal{G}_{\{H''\}}$, for all $H'' \in \mathcal{H}$. Finally, from Property 1.(e), we have that $H' \notin \mathcal{G}_{\mathcal{H}'}$ and $H' \in \mathcal{G}_{\mathcal{H}}$ and therefore that $\mathcal{G}_{\mathcal{H}} \neq \mathcal{G}_{\mathcal{H}'}$. □

Proposition 10 *Let $\mathcal{F}$ be a class of graphs. If $\mathcal{F}$ has an FSC, then the unique minimal FSC of $\mathcal{F}$ is* $\min(\overline{\mathcal{F}})$.

Proof. If $\mathcal{F}$ has an FSC, we know from Proposition 8 that it is hereditary and that $\overline{\mathcal{F}}$ is an FSC of $\mathcal{F}$. Moreover, by definition, we know that $\min(\overline{\mathcal{F}})$ is also an FSC of $\mathcal{F}$ that is minimal. Finally, we know from Proposition 9 that $\mathcal{F}$ has a single minimal characterization, which can only be $\min(\overline{\mathcal{F}})$. □

Consider an hereditary class of graphs $\mathcal{F}$ and let G be a graph in $\overline{\mathcal{F}}$. Furthermore, let $G' = G - \{v\}$ be a graph produced by removing from G any vertex $v \in V(G)$ and its incidents edges. If $G' \in \mathcal{F}$, since $\mathcal{F}$ is hereditary, all subgraphs of G' are also in $\mathcal{F}$. Therefore, we know that a graph G is minimal in $\overline{\mathcal{F}}$ if all its subgraph produced by removing a vertex are in $\mathcal{F}$. This is the main principle behind our algorithm to find FSCs. At each iteration k, we find a graph $G_k \in \overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M}_k}$, where $\mathcal{M}_0 = \emptyset$. Then, we search for a vertex v of G_k such that $G_k - \{v\}$, is still in $\overline{\mathcal{F}}$. If no such vertex exists, then G_k is minimal and we add it to $\mathcal{M}_k$. Otherwise, we remove v from G_k and look for another vertex with the same properties. We repeat this process until $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M}_k} = \emptyset$. The problem with this simple approach is that the FSC can contain an infinite number of graphs. Thus, to have an algorithm that stops in a finite number of steps, we need to impose a limit on the number of graphs allowed in the FSC. Let N be the maximum number of graphs in the FSC, the algorithm is summarized in the following procedure:

1. Let $\mathcal{M}_0 := \emptyset$ and $k := 0$;
2. If $\exists G_k \in \overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{M}_k}$ and $k \leq N$ then
   2.1. While $\exists v \in V(G_k)$ s.t. $G_k - \{v\} \in \overline{\mathcal{F}}$ do $G_k := G_k - \{v\}$;
   2.2. Let $\mathcal{M}_{k+1} := \mathcal{M}_k \cup \{G_k\}$;
   2.3. Let $k := k + 1$ and goto (*2*);
3. Else, return $\mathcal{H} = \mathcal{M}_k$;

Proposition 11 *The above algorithm returns, in a finite number of steps, the set* $\mathcal{H} = \min(\overline{\mathcal{F}})$*, if* $|\min(\overline{\mathcal{F}})| \leq N$*, or a set* $\mathcal{H}$ *containing* N *graphs of* $\min(\overline{\mathcal{F}})$*.*

Proof. Let G_i and G_j be two graphs, such that $i < j$. Since $G_i \in \mathcal{M}_j$ and $G_j \in \mathcal{G}_{\mathcal{M}_j}$, we have that $G_i \not\subseteq G_j$ and therefore that $G_i \neq G_j$. Moreover, since, at every iteration k, we have $|\mathcal{M}_k| = k$, the algorithm will terminate in at most N steps. Furthermore, let G_k be any graph of $\mathcal{H}$. Suppose that G_k is not minimal, then there is a vertex $v \in V(G_k)$ such that $G_k - \{v\} \in \overline{\mathcal{F}}$, which is a contradiction since G_k is added to $\mathcal{H}$ only if no such vertex exists. Therefore, $\mathcal{H} \subseteq \min(\overline{\mathcal{F}})$. Finally, we know that the algorithm terminates only when $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{H}} = \emptyset$ or $|\mathcal{H}| = N$. In the first case, we know that $\mathcal{H}$ is an FSC of $\mathcal{F}$, and since it is minimal, we know from Proposition 9 that it is unique, and thus $\mathcal{H} = \min(\overline{\mathcal{F}})$. Otherwise, $\mathcal{H}$ contains N graphs of $\min(\overline{\mathcal{F}})$. $\square$

6 Automated conjecture generation

In the last sections, we have presented some exact methods to find FSCs of a class of graphs $\mathcal{F}$, or conditions to have an FSC. Although these methods guarantee to find FSCs, SFSCs or NFSCs of $\mathcal{F}$, if they exist, this might be impossible in practice. Thus, in many of our algorithms, we need to determine whether or not the sets $\mathcal{F} \cap \overline{\mathcal{G}}_{\mathcal{H}}$ and $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{H}}$ are empty, where $\mathcal{H}$ is a non-empty set of graphs. Since $\mathcal{F}$ and $\overline{\mathcal{F}}$ can be infinite, this may very well be an undecidable problem. To overcome this problem, we impose some restrictions on the graphs considered by our algorithms. Let $\mathcal{G}^n$ denote the graphs of $\mathcal{G}$ that have at most n vertices. We limit the graphs considered by our algorithms to $\mathcal{G}^n$, and use $\mathcal{G}^n \cap \mathcal{F}$ and $\mathcal{G}^n \cap \overline{\mathcal{F}}$ instead of $\mathcal{F}$ and $\overline{\mathcal{F}}$. Consequently, the guarantee that the results obtained by our algorithm are FSCs or conditions to have an FSC no longer holds. However, we can still use these algorithms to generate conjectures based on the hypothesis that if these results are true for graphs of n or less vertices, then they must be true for all graphs.

Limiting the order of graphs considered by our algorithms makes finding graphs of a given class practicable. Yet, finding these graphs is still a difficult task. Let $\mathcal{C}$ be a given class of graphs, and n be a given integer, we are interested in finding a graph of $\mathcal{C}$ that has at most n vertices. In the next sections, we present two approaches for this task: an enumerative approach and a heuristic one.

6.1 Enumerative approach

In the enumerative approach, we generate one by one the graphs of $\mathcal{G}^n$ until we find a graph in $\mathcal{C}$. While this approach might seem somewhat inefficient, it is practical for two reasons. First, it guarantees to find a graph in $\mathcal{G}^n \cap \mathcal{C}$, if one exists. Second, although the number of unlabeled graphs increases exponentially with their maximum order, the majority of results on forbidden subgraphs in the literature involve graphs having 10 or less vertices, which can be exhaustively enumerated quite rapidly.

To enumerate the graphs of n or less vertices, we use the method proposed by McKay [13]. This method employs a canonical representation of a graph G, which partially orders the vertices of G in a way that is independent of their label. This partial ordering is then used to transform the lattice of graphs into a rooted tree with a function p that uniquely maps a graph G to a parent graph $p(G)$. The root of this tree is the graph containing a single vertex, and the parent $p(G)$ of G is obtained by removing from G a minimum vertex according to the partial ordering. By traversing this tree in a depth-first manner, we can thus enumerate the graphs without generating isomorphic copies. An important advantage of this approach is that is allows to efficiently prune the search space. To illustrate this, suppose we want to find a graph of $\mathcal{C} = \overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{H}}$, and let G be any graph. If $G \notin \mathcal{G}_{\mathcal{H}}$, then there is a graph $H \in \mathcal{H}$ such that $H \subseteq G$. Since all supergraphs[1] of G will also contain H, we can avoid the exploration of the subtree rooted at G without fear of missing any graphs of $\mathcal{C}$.

6.2 Heuristic approach

It some cases, the FSCs and conditions to have an FSC for a graph class $\mathcal{C}$ may be out of reach of the enumerative approach. We then need to trade the completeness of that approach to gain the ability of finding graphs that have more vertices. This sacrifice is a reasonable one, since we already limit the order of graphs, and thus, cannot do better than find conjectures.

The heuristic approach we use is similar to the one employed by AGX to find extremal graphs [4]. In this approach, the task of finding a graph of $\mathcal{C}$ is modeled as an optimization problem where the objective function f determines the membership of a graph G in $\mathcal{C}$. For example, we can express this function such that $f(G) \leq 0$ if and only if G belongs to $\mathcal{C}$, and use any heuristic search method to minimize f until we find $f(G) \leq 0$. In AGX, the heuristic used is the *Variable Neighborhood Search* [14]. This heuristic starts with a random graph $G \in \mathcal{G}^n$ and explores the neighborhood containing the graphs that can be obtained from G by considering every possible group of 2 to 4 vertices, and then adding or removing edges between vertices of this group. If no neighbor graph improving G is found, the neighborhood is then widened by increasingly adding a random perturbation to G before exploring its neighborhood, until a neighbor improving G is found, or until the maximum neighborhood size is attained or a given time limit is reached. Furthermore, if

[1] A graph G is a supergraph of a graph H if and only if H is a subgraph of G.

a neighbor improving G is found, the search is then recentered on this neighbor and the original neighborhood is restored. Using this approach, AGX was able to find extremal graphs of up to 50 vertices.

To illustrate this approach, consider a set of graphs invariants $\{\gamma_1, \gamma_2, \ldots, \gamma_M\}$, and a class containing graphs G such that

$$a_1\gamma_1(G) \ + \ a_2\gamma_2(G) \ + \ \ldots \ + \ a_M\gamma_M(G) \ \leq \ b,$$

where $a_i \in \mathbb{R}$, $1 \leq i \leq M$ and $b \in \mathbb{R}$. Furthermore, let $\mathcal{H} = \{H_1, H_2, \ldots, H_N\}$ be a set of forbidden subgraphs, and denote $\mathrm{emb}(H, G)$ the number of different embeddings, i.e. subgraph isomorphisms, of H in G. The task of finding G in $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{H}}$ can be formulated as maximizing

$$f(G) = -b \ + \ \sum_{i=1}^{M} a_i\gamma_i(G) \ - \ B\sum_{j=1}^{N} \mathrm{emb}(H_j, G),$$

where B is big constant chosen such that any non-zero value of the right sum dominates the value of the left sum minus b. If we find G such that $f(G) > 0$, we know that G is in $\overline{\mathcal{F}} \cap \mathcal{G}_{\mathcal{H}}$.

7 Experimental results

In this section, we use our algorithms to find results related to the concepts of independence, domination and irredundance of graphs. Before presenting the results, we need to define these concepts.

An *independent* (or *stable*) set S is a set of pairwise non-adjacent vertices. The *independence number* of a graph G, written $\alpha(G)$, is the maximum cardinality of an independent set of G, and the *independent domination number* of G, denoted $i(G)$, is the minimum cardinality of a maximal independent set of G. Furthermore, a *dominating set* T is a set of vertices such that each vertex of $V(G) \setminus T$ is adjacent to at least one vertex of T. The *domination number* of G, written $\gamma(G)$, is the minimum cardinality of a dominating set of G, and we denote $\Gamma(G)$ the maximum cardinality of a minimal dominating set of G. Moreover, let $X \subseteq V$, a vertex $x \in X$ is irredundant in X if it is isolated in X or if it has a private neighbor, i.e. a vertex $y \in V \setminus X$ such that x is the only vertex of X adjacent to y. The set X is irredundant if all its vertices are irredundant. We denote $IR(G)$ and $ir(G)$, respectively, the maximum cardinality of an irredundant set of G and the minimum cardinality of a maximal irredundant set of G. A famous result known as the *domination chain*, e.g. [12], states that the following relations hold for all graphs $G \in \mathcal{G}$:

$$ir(G) \leq \gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G) \leq IR(G).$$

In the next two sections, we search for SFSCs and NFSCs on the classes of graphs that satisfy or not some of these relations at equality. This is a two step process. First, we

use our algorithms to find conjectures. Working under the assumption that the forbidden subgraphs have at most 10 vertices, we use the enumerative approach to find specific graphs in these algorithms. We then demonstrate by hand the conjectures found by our algorithms.

7.1 Conjectures on SFSC

As a first experiment, we set out to find SFSCs for the classes of graphs which satisfy at equality the relations of the domination chain. This task has been the subject of previous work by graph theorists. In [1], Allan and Laskar have shown that if a graph G does not contain the first graph of Figure 2, known as the "claw", as subgraph, then $\gamma(G) = i(G)$. Furthermore, in [7], Favaron, based on some earlier work by Bollobas and Cockayne [3], conjectured that if G does not contain as subgraph any of the graphs of Figure 3, then $ir(G) = \gamma(G)$. This conjecture was later proved by Puech in [15], and by Volkmann and Zverovich in [17]. Favaron also showed in [7] that if G does not contain the first nor the second graph of Figure 2, known as the "deer", then $ir(G) = i(G)$, and that if G does not contain any graph of the same figure, then $\Gamma(G) = IR(G)$.

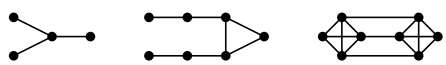


Figure 2: Graphs used in the SFSCs for graphs satisfying at equality some relations of the domination chain.

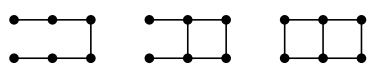
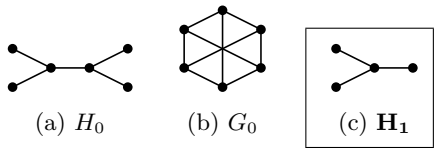


Figure 3: A SFSC for the class of graphs G such that $ir(G) = \gamma(G)$.

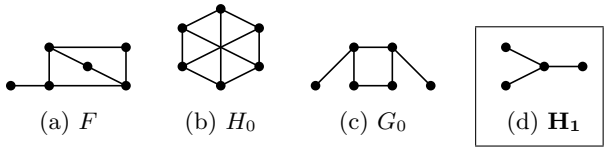
We first generated conjectures on SFSCs for the class of graphs G such that $\gamma(G) = i(G)$. Figure 4 shows a particular execution of our algorithm using $N = 1$ as maximum number of graphs and $\mathcal{G}^{10}$ as search space. At iteration $k = 0$, the algorithm finds a graph H_0 such that $\gamma(H_0) < i(H_0)$ and a graph G_0 that does not contain H_0 and such that $\gamma(G_0) < i(G_0)$. The algorithm then finds a graph H_1 that is a common subgraph of both H_0 and G_0. Since the algorithm cannot find an H_1-free graph G_1 such that $\gamma(G_1) < i(G_1)$, it return $\mathcal{H} = \{H_1\}$, which is the SFSC proposed by Allan and Laskar in [1].

Note that we can obtain the same SFSC for higher values of N, as shown in Figure 5. In this example, where $N = 2$ is used, the algorithm first finds a graph $F \in \overline{\mathcal{F}}$ of minimum order, such that $\gamma(F) < i(F)$, and then searches for a maximum common subgraph of $\overline{\mathcal{F}} \cap G_{\{F\}}$. Since the algorithm cannot find a graph G_1 that does not contain F or H_1, and such that $\gamma(G_1) < i(G_1)$, it stops with $\{F, H_1\}$ as SFSC. However, since $H_1 \subset F$, the algorithm removes F from the SFSC and, once more, returns $\mathcal{H} = \{H_1\}$.



(a) H_0 (b) G_0 (c) $\mathbf{H_1}$

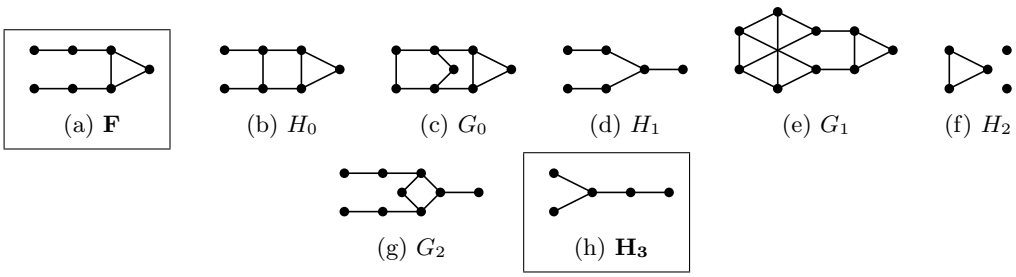
Figure 4: An illustration of the SFSC algorithm, for the class of graphs G such that $\gamma(G) = i(G)$, using $N = 1$ as maximum number of graphs and $\mathcal{G}^{10}$ as search space. Graph H_1 is an SFSC that was originally found by Allan and Laskar in [1].



(a) F (b) H_0 (c) G_0 (d) $\mathbf{H_1}$

Figure 5: Another illustration of the SFSC algorithm, for the class of graphs G such that $\gamma(G) = i(G)$, using $N = 2$ as maximum number of graphs and $\mathcal{G}^{10}$ as search space. Graph H_1 is an SFSC that was originally found by Allan and Laskar in [1].

We have also used our algorithm to find conjectures on SFSCs for the class of graphs G such that $ir(G) = \gamma(G)$. In the execution shown in Figure 6, our algorithm generated the conjecture that the graphs F and H_3 of Figure 6, known as the "fork" and the "deer', form an SFSC for this class of graphs. As we now prove, this conjecture is a novel result that strengthens Favaron's result, presented in [7], that $ir(G) = \gamma(G)$ if G is "claw"-free and "deer"-free.



(a) $\mathbf{F}$ (b) H_0 (c) G_0 (d) H_1 (e) G_1 (f) H_2

(g) G_2 (h) $\mathbf{H_3}$

Figure 6: An illustration of the SFSC algorithm, for the class of graphs G such that $ir(G) = \gamma(G)$, using $N = 2$ as maximum number of graphs and $\mathcal{G}^{10}$ as search space. Graphs F and H_3 form a novel SFSC that strengthens a previous result proposed by Favaron in [7].

Theorem 1 *If a graph G does not contain the graph F or H_3 of Figure 6, known as the "fork" and the "deer", then $ir(G) = \gamma(G)$.*

Proof. Let I be an irredundant set with $|I| = ir(G)$, and let A be the set of vertices $a \in V \setminus I$ such that $N_I(a) = \emptyset$. If A is empty then I is a dominating set and we have $\gamma(G) \leq ir(G)$. So, we assume that $A \neq \emptyset$ and denote $N_X(v)$ the set of vertices in X that are adjacent to a vertex v. Furthermore, for a vertex $x \in I$, we denote $P(x)$ the subset of vertices $y \notin I$ such that $N_I(y) = \{x\}$. Since I is irredundant, $P(x) \neq \emptyset$ for all $x \in I$ such that $N_I(x) \neq \emptyset$. Let $W \subseteq I$ be the subset of vertices in I which are redundant in $I \cup A$, and for a vertex $w \in W$, let $A(w)$ be the subset of vertices in A such that w is redundant in $I \cup \{a\}$. Note that $\bigcup_{w \in W} A(w) = A$ and $A(w) \neq \emptyset$ for every $w \in W$. Also, every vertex $a \in A(w)$ is adjacent to every vertex $u \in P(w)$ for all $w \in W$, else w would be irredundant in $I \cup \{a\}$ Moreover, $P(w)$ is a clique for all $w \in W$, else there exist $w \in W$ and two non adjacent vertices u_1 and u_2 in $P(w)$, and it follows that $I \cup \{u_1\}$ is irredundant (since any vertex in $A(w)$ is a private neighbor of u_1 while u_2 is a private neighbor of w), a contradiction.

We now prove that every connected component of the subgraph $G[W]$ is a clique. This is obviously true for connected components with at most 2 vertices. So consider any connected component with at least three vertices, and assume it is not a clique. Then there are three vertices w_1, w_2 and w_3 in this component such that w_2 is adjacent to w_1 and w_3 while w_1 and w_3 are not adjacent. But then $\langle w_1, w_2, w_3, u, a \rangle \simeq H_3$ for all $u \in P(w_2)$ and $a \in A(w_2)$.

The next observation is that the vertices in $N_A(u)$ induce a clique in $G[w]$ for all vertices $u \in \bigcup_{w \in C} P(w)$. Indeed, assume there exist $w \in W$, a vertex $u \in P(w)$, and two non adjacent vertices a_1 and a_2 in $N_A(u)$. Then, since $N_I(w) \neq \emptyset$, there is a vertex $x \in I$ adjacent to w, which means that $\langle w, x, u, a_1, a_2 \rangle \simeq H_3$, a contradiction. As a consequence, $A(w)$ is a clique for all $w \in W$.

Note also that if C is the vertex set of a connected component of $G[W]$ with at least 3 vertices, then $\bigcup_{w \in C} A(w)$ induces a clique in G. Indeed, if $\bigcup_{w \in C} A(w)$ does not induce a clique, then there are two vertices w_1 and w_2 in C and two non adjacent vertices a_1 and a_2 such that a_1 belongs to $A(w_1)$ but not to $A(w_2)$, while a_2 belongs to $A(w_2)$ but not to $A(w_1)$. Consider any $u_1 \in P(w_1)$ and $u_2 \in P(w_2)$. Since $N_A(u_1)$ and $N_A(u_2)$ both induce a clique in $G[W]$, we know that u_1 is not adjacent to a_2 and u_2 is not adjacent to a_1. Vertices u_1 and u_2 are not adjacent else $\langle w_1, u_1, u_2, a_1, a_2 \rangle \simeq H_3$. Then, for any $w_3 \in C \setminus \{w_1, w_2\}$ we have $\langle w_1, w_2, w_3, u_1, u_2, a_1, a_2 \rangle \simeq F$, a contradiction.

The connected component of $G[W]$ with one vertex will be said to be of type 1, while those with at least 3 vertices will be said to be of type 2. We split the connected components with 2 vertices w_1, w_2 into three groups.

- If there exist a vertex in $P(w_1) \cup P(w_2)$ adjacent to all vertices in $A(w_1) \cup A(w_2)$, then the component is said to be of type 3.
- If every vertex in $P(w_1) \cup P(w_2)$ has a non neighbor in $A(w_1) \cup A(w_2)$, and there exist two non adjacent vertices $u \in P(w_i)$ and $a \in A(w_j)$ $(j \neq i)$ such that a is adjacent to all vertices in $A(w_i)$, then the component is said to be of type 4.

- If every vertex in $P(w_1) \cup P(w_2)$ has a non neighbor in $A(w_1) \cup A(w_2)$, and for all non adjacent vertices $u \in P(w_i)$ and $a \in A(w_j)$ $(j \neq i)$ there exist a vertex a' not adjacent to a, then the component is said to be of type 5.

We now create a set I' from I as follows.

- For every connected component $C = \{w\}$ of type 1, we remove w from I and add a vertex $u \in P(w)$.
- For every connected component C of type 2, we remove a vertex $w \in C$ from I and add a vertex $a \in A(w)$.
- For every connected component $C = \{w_1, w_2\}$ of type 3, one of the vertices in C, say w_1, has a neighbor $u \in P(w_1)$ which is adjacent to all vertices in $A(w_1) \cup A(w_2)$. We remove w_1, from I and replace it by u.
- For every connected component $C = \{w_1, w_2\}$ of type 4, there exist two non adjacent vertices $u \in P(w_i)$ and $a \in A(w_j)$ $(j \neq i)$ such that a is adjacent to all vertices in $A(w_i)$. We remove w_j from I and replace it by a.
- For every connected component $C = \{w_1, w_2\}$ of type 5, we remove w_1 and w_2 from I and replace them by a vertex $u_1 \in P(w_1)$ and a vertex $u_2 \in P(w_2)$.

Since $|I'| = |I|$ it is now sufficient to prove that I' is a dominating set. So let v be a vertex that does not belong to I'.

- If $v \in I$ then v was removed from a connected component C of one of the above five types. If C is of type 1 or 5, then v is adjacent to the neighbor u of v that was added to I'. Else, C contains at least one neighbor w of v that was not removed from I.
- If $v \in \bigcup_{w \in W} P(w)$ then let w be the vertex in W such that $v \in P(w)$. If $w \in I'$, then w is a neighbor of v in I'. If $w \notin I'$, then w was removed from a connected component C. If C is of type 1, 3 or 5, then w was replaced by a vertex in $P(w)$, and v has a neighbor in I' since $P(w)$ is a clique. If C is of type 2 or 4, then w was replaced by a vertex in $A(w)$, and v has a neighbor in I' since v is adjacent to all vertices in $A(w)$.
- If $v \in A$ then let w be a vertex in W such that $v \in A(w)$ and let C be the connected component of $G[W]$ containing w. If C is of type 1 or 3, then a vertex w was replaced by a vertex $u \in P(w)$, and v has a neighbor in I' since u is adjacent to all vertices in $\bigcup_{x \in C} A(x)$. If C is of type 2 or 4, then a vertex in C was replaced by a vertex $a \in \bigcup_{x \in C} A(x)$, and v has a neighbor in I' since a is adjacent to all other vertices in $\bigcup_{x \in C} A(x)$. Finally, if C is of type 5, then the vertices w_1 and w_2 have been replaced by $u_1 \in P(w_1)$ and $u_2 \in P(w_2)$, which means that v has a neighbor in I' since $v \in N_A(u_1) \cup N_A(u_2)$.
- If $v \notin (I \cup A \cup \bigcup_{w \in W} P(w)$ then v is adjacent to at least two vertices in I. If a neighbor of v belongs to I' then there is nothing to prove. So assume v has no neighbor in I' and consider any neighbor $w \in I \setminus I'$ of v. Vertex w belongs to a connected component C of $G[W]$ and was removed from I.

- If C is of type 1 or 3, then w was replaced by a vertex $u \in P(w)$. Let a be any vertex in $A(w)$. Vertex v is not adjacent to a else $I \cup \{v\}$ is irredundant. Since $N_I(w) \neq \emptyset$, there is a vertex $w' \in I$ adjacent to w. Note that $w' \in I'$ since no neighbor of w is removed from I. Hence, v is not adjacent to w'. If v is adjacent to u then v has a neighbor in I'. Otherwise, $\langle v, w, w', u, a\rangle \simeq H_3$, a contradiction.
- If C is of type 2, then let x be any vertex in $I \setminus \{w\}$ adjacent to v. Such a vertex necessarily exists since $v \notin P(w)$. Since $x \notin I'$ while no neighbor of w is removed from I, there exists a connected component $C' \neq C$ of $G[W]$ with $x \in C'$. Consider any two vertices w' and w'' in $C \setminus \{w\}$, and let u be any vertex in $P(w')$ and a any vertex in $A(w')$. Vertex v is not adjacent to u else $\langle v, w, w'', u, x\rangle \simeq H_3$. Also, v is not adjacent to a else $\langle v, w, w', x, a\rangle \simeq H_3$. Hence $\langle v, w, w', w'', x, u, a\rangle \simeq F$, a contradiction.
- If C is of type 5, then the vertices w_1 and w_2 were replaced by $u_1 \in P(w_1)$ and $u_2 \in P(w_2)$. Since no vertex in $P(w_2)$ is adjacent to all vertices in $A(w_1)$, consider a vertex $a_1 \in A(w_1)$ not adjacent to u_1. Since a_1 is not adjacent to all vertices in $A(w_2)$, consider any vertex $a_2 \in A(w_2)$ not adjacent to a_1. Vertex v is not adjacent to a_1 nor to a_2 else $I \cup \{v\}$ would be irredundant. Also, we may assume that v is not adjacent to u_1 nor to u_2 else v has a neighbor in I'. Hence, u_1 is not adjacent to a_2 else $\langle w_1, w_2, u_1, a_1, a_2\rangle \simeq H_3$, and u_1 is not adjacent to u_2 else $\langle w_1, u_1, u_2, a_1, a_2\rangle \simeq H_3$. Since w is one of the two vertices w_1 or w_2, we know that v is adjacent to at least one of them, say w_1. But then $\langle v, w_1, w_2, u_1, a_1\rangle \simeq H_3$ (if v is not adjacent to w_2) or $\langle v, w_1, w_2, u_1, u_2, a_1, a_2\rangle \simeq F$, a contradiction.
- If C is of type 4, then let x be any vertex in $I \setminus \{w\}$ adjacent to v, and let $C' \neq C$ be the connected component of $G[W]$ with $x \in C'$. We have shown above that if C' is of type 1, 2, 3 or 5, then v has a neighbor in I'. So assume C' is also of type 4. Let w' be the second vertex in C, let x' the second vertex in C', and let u be any vertex in $P(w)$. Note that $\{w', x'\} \subseteq I'$ since no neighbor of w or x is removed from I, and it follows that v is not adjacent to w' and x'. Since C is of type 4, w was replaced by a vertex a adjacent to all vertices in $A(w')$, and not adjacent to a vertex $u' \in P(w')$. Since no vertex in $P(w)$ is adjacent to all vertices in $A(w')$, consider any $a' \in A(w')$ not adjacent to u. If v is adjacent to a then v has a neighbor in I'. So assume v and a are not adjacent. Vertex v is then necessarily adjacent to u else $\langle v, w, w', u, a\rangle \simeq H_3$, v is not adjacent to u' else $\langle v, w, u', x, x'\rangle \simeq H_3$, and u is not adjacent to u' else $\langle v, u, u', x, a\rangle \simeq H_3$. But then $\langle v, w, w', x, x', u, u'\rangle \simeq F$, a contradiction.

□

If we combine the SFSCs we obtained for the classes $ir(G) = \gamma(G)$ and $\gamma(G) = i(G)$, we get the SFSC proposed by Favaron: if a graph G is "claw"-free and "deer"-free then

$ir(G) = i(G)$. However, if the SFSC algorithm randomly selects the maximum common subgraphs, at each iteration, we can obtain a completely different SFSC. For instance, in the execution shown in Figure 7, our algorithm found a path of 5 vertices, which was shown by Puech in [15] to be an SFSC.

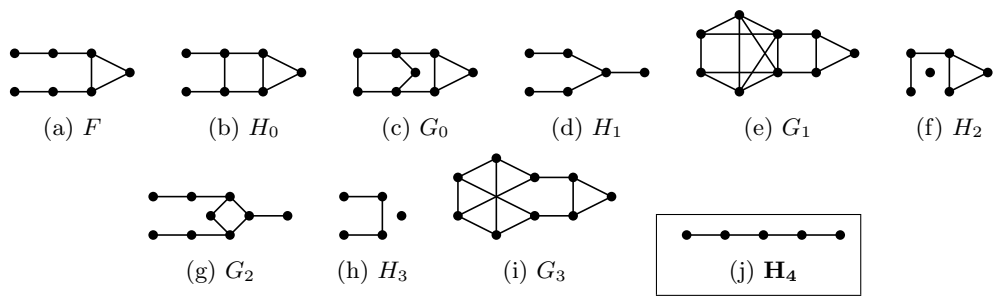


Figure 7: Another illustration of the SFSC algorithm, for the class of graphs G such that $ir(G) = \gamma(G)$, using $N = 2$ as maximum number of graphs and $\mathcal{G}^{10}$ as search space. Graph H_4 was shown be an SFSC by Puech in [15].

7.2 Conjectures on NFSC

Following what we have done for SFSCs, we look in this section for NFSCs on the classes of graphs satisfying at inequality some of the relations of the domination chain.

As a first experiment, we used our algorithm to find conjectures on NFSCs for the class of graphs G such that $\gamma(G) < i(G)$. Figure 8 shows a particular execution of the algorithm for this task, using $\mathcal{G}^8$ as search space. At iteration $k = 0$, the algorithm finds a graph G_0

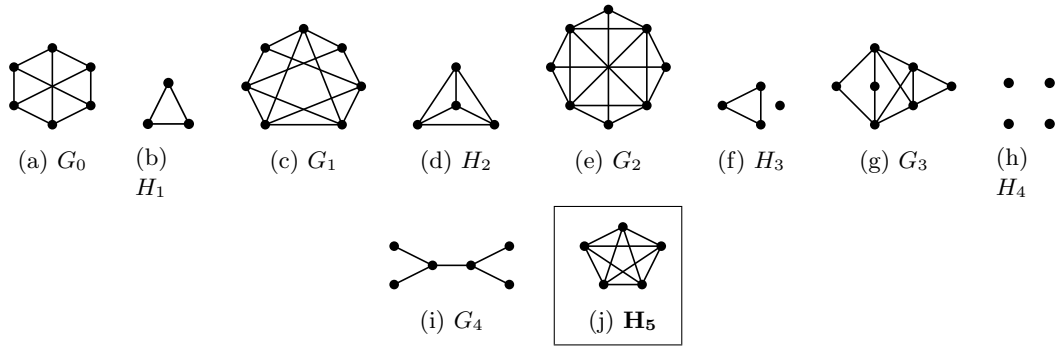


Figure 8: An illustration of the NFSC algorithm, for the class of graphs G such that $\gamma(G) < i(G)$, using $\mathcal{G}^8$ as search space.

such that $\gamma(G_0) < i(G_0)$. Since $\mathcal{L}_1$ only contains G_0, the algorithm then finds a minimum

order graph H_1 not included in G_0. At the next iteration, the algorithm then finds a graph G_1 containing H_1 and such that $\gamma(G_1) < i(G_1)$, and a graph H_2 of minimum order that is not a subgraph of G_0 or G_2. This process is repeated until the algorithm reaches iteration $k = 5$, where it finds a 5 vertex clique H_5, as the minimum order graph not included in $\{G_0, G_1, G_2, G_3, G_4\}$. However, the algorithm does not find a graph G_5 containing H_5 and such that $\gamma(G_5) < i(G_5)$, and thus returns $\mathcal{H} = \{H_5\}$. Repeating this experiment using $\mathcal{G}^9$ and $\mathcal{G}^{10}$ as search space, our algorithm has found complete graphs of 6 and 7 vertices, suggesting the novel NFSC that a graph G of n vertices is $K_{(n-3)}$-free if $\gamma(G) < i(G)$. We now demonstrate this result.

Theorem 2 *Let G be any graph of order n. If $\gamma(G) < i(G)$ then G is $K_{(n-3)}$-free.*

Proof. Similar to what we did for the previous two theorems, we prove this by showing that a minimum cardinality dominating set can be transformed into an independent dominating set of the same cardinality. Since $i(G) \geq \gamma(G)$ for all graphs, we only need to prove that $i(G) \leq \gamma(G)$.

Let $D = \{w_1, w_2, \ldots, w_m\}$ be a minimum cardinality dominating set of G. If $E(G[D]) = \emptyset$ then D is an independent set, and we have $i(G) \leq \gamma(G)$. Otherwise, let (w_1, w_2) be any edge of $E(G[D])$, and let $P(w)$ be the private neighbors of w. We know that $P(w_i) \neq \emptyset$, $i = 1, 2$, otherwise $D' = D \setminus \{w_i\}$ would be a dominating set of one less vertex than D, which contradicts the minimality of D. We show that at least one of $P(w_1)$ or $P(w_2)$ is a complete graph. Suppose this is not the case, then there are four vertices u_1, v_1, u_2 and v_2, such that u_1, v_1 are non-adjacent private neighbors of w_1, and u_2,v_2 are non-adjacent private neighbors of w_2. However, $\langle u_1, v_1, w_1, u_2, v_2, w_2\rangle$ contains no triangle, and thus G contains no $K_{(n-3)}$, which is a contradiction. Therefore, $P(w_1)$ or $P(w_2)$ forms a complete graph, without loss of generality, suppose it is $P(w_1)$. Let u_1 be any vertex of $P(w_1)$. The set $D' = (D \setminus \{w_1\}) \cup \{u_1\}$ is a dominating set of cardinality $|D'| = |D|$ such that $E(G[D']) < E(G[D])$. By repeating this process iteratively at most $n - 3$ times, we thus get a dominating set of cardinality $|D|$, which is also an independent set. Consequently, we have $i(G) \leq \gamma(G)$. $\square$

In the next experiment, we searched for NFSCs on the class of graphs G such that $ir(G) < \gamma(G)$. Figure 9 shows an execution of our NFSC algorithm, using $\mathcal{G}^8$ as search space, where a complete graph of 5 vertices is found. Again, when repeating this experiment with G^9 and G^{10}, our algorithm found, although not systematically, complete graphs of 6 and 7 vertices, which suggested a new theorem and corollary:

Theorem 3 *Let G be any graph of order n. If $ir(G) < \gamma(G)$ then G is $K_{(n-3)}$-free.*

Proof. Let I be a minimum cardinality maximal irredundant set of G, and let A be the set of vertices not dominated by I. If $A = \emptyset$, then I is a dominating set, and we have $\gamma(G) \leq ir(G)$. Otherwise, consider any $a \in A$. There is a non-isolated vertex $w \in I$

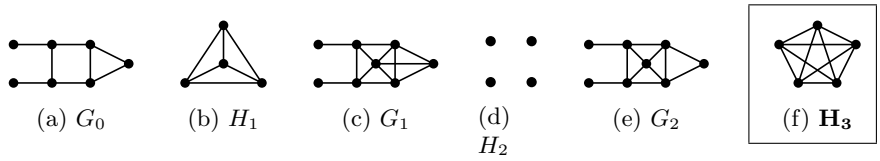


Figure 9: An illustration of the NFSC algorithm, for the class of graphs G such that $ir(G) < \gamma(G)$, using $\mathcal{G}^8$ as search space.

which is redundant in $I \cup \{a\}$, else I would not be maximal. Let $P(w)$ denote the private neighbors of w. We know that $P(w)$ is not empty since w is non-isolated in I. Furthermore, we know that a is adjacent to all the vertices of $P(w)$, otherwise w would be irredundant in $I \cup \{a\}$. We also know that $P(w)$ forms a complete graph, otherwise let u_1 and u_2 be two private neighbors of w, $I \cup \{u_1\}$ is irredundant, since u_1 has a as private neighbor and w the vertex u_2, thus contradicting the maximality of I. Denote $A(w) \subseteq A$ the vertices a such that w is redundant in $I \cup \{a\}$, and let W be the vertices of I that are redundant in $I \cup A$. We show that I can be transformed into a dominating set I' having no more vertices then I, and thus that $\gamma(G) \leq ir(G)$.

Consider any vertex $w_1 \in W$ such that $|A(w_1)|$ is maximum. We show that $A(w_1) = A$. Suppose this is not the case, then consider any vertex $a_2 \in A \setminus A(w_1)$, and let w_2 be a vertex redundant in $I \cup \{a_2\}$. Since a_2 belongs to $A(w_2)$ but not to $A(w_1)$, and since $|A(w_1)| \geq |A(w_2)|$, there is a vertex $a_1 \in A(w_1)$ that does not belong to $A(w_2)$. Let u_1 be a vertex in $P(w_1)$ that is not adjacent to a_2, and u_2 be a vertex in $P(w_2)$ that is not adjacent to a_1. However, $\langle a_1, u_1, w_1, a_2, u_2, w_2 \rangle$ contains no triangle, and thus G contains no $K_{(n-3)}$, which is a contradiction. Therefore, $A(W_1) = A$, and $I' = (I \setminus \{w_1\}) \cup \{u_1\}$ is a dominating set of cardinality $|I'| \leq |I|$. $\square$

Corollary 1 *Let G be any graph of order n. If $ir(G) < i(G)$ then G is $K_{(n-3)}$-free.*

7.3 Conjectures on FSC

Hereditary graph classes that can be characterized with a limited number of forbidden subgraphs are not that common in graph theory. Furthermore, those that are known have been studied in depth, and have already been characterized with forbidden subgraphs. Thus, rather than searching for new FSCs, we will, in this section, try to reproduce some known results.

A *split graph* is a graph in which the vertices can be partitioned into a clique and an independent set. This class of graphs has some very interesting properties that have been studied, among others, by Földes, Hammer and Simeone [8, 10]. One of these properties is that split graphs can be recognized in linear time, using the sequence of degrees of each vertex. Let G be graph with n vertices, we order the degrees of the vertices of G, by non-increasing value, to form a sequence $d_1 \geq d_2 \geq \ldots \geq d_n$. Let m be the largest value

of i such that $d_i \geq i-1$, then G is a split graph if and only if

$$m(m-1) \; - \; \sum_{i=1}^{m} d_i \; + \; \sum_{j=m+1}^{n} d_j \; \leq \; 0.$$

If this is the case, the m vertices with the largest degrees form a maximum clique in G, and the remaining vertices an independent set.

Using the above test to determine whether or a graph is a split graph and $\mathcal{G}^8$ as search space, we have found the minimal FSC for this class, shown in Figure 10, containing two cycles of 4 and 5 vertices, and a graph made of two disjoint edges. This FSC is a well known result, first described by Földes and Hammer in [8].

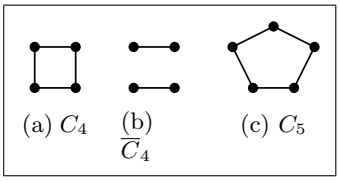


Figure 10: An illustration of the FSC algorithm, for the class of split graphs, using $\mathcal{G}^8$ as search space.

8 Conclusion

We have presented, in this paper, some methods to automate the discovery of new conjectures on forbidden subgraph characterization. The first two algorithms that were described allow to find sufficient conditions to have a characterization, in the form of a set of graphs $\mathcal{H}$ such that a graph G is in a graph class $\mathcal{F}$ if G is $\mathcal{H}$-free. While the first of these algorithms returns a single forbidden subgraph, the second allows to find sufficient conditions involving multiple subgraphs. Since a given graph class can have a great number of sufficient conditions, we described two measures, complexness and tightness, that were used to select the forbidden subgraphs. Furthermore, using these same measures, we have developed two other algorithms to find necessary conditions in the form of a set of graphs $\mathcal{H}$ such that all graphs of a given class $\mathcal{F}$ are $\mathcal{H}$-free. Lastly, we presented an algorithm that finds actual characterizations.

Although these algorithms find forbidden subgraph characterizations, or conditions to have such characterizations, we have seen that this can be impossible in practice, due to the infinite number of graphs. However, we have shown that, by bounding the order of graphs under consideration, we could use our algorithms to find conjectures. Following this idea, we have used our algorithms to reproduce some important results, as well as to find new ones, on the domination, independence and irredundance of graphs. In particular, our algorithms conjectured two new results which we proved were correct: a n vertex graph G

is such that $ir(G) = \gamma(G)$ if it is "fork"-free and "deer"-free, and that G is $K_{(n-3)}$-free if $ir(G) < i(G)$.

References

[1] R. Allan and R. Laskar. On domination and independent domination numbers of a graph. *Discrete Mathematics*, 23:73–76, 1978.

[2] L. Beineke. Characterizations of derived graphs. *Journal of Combinatorial Theory*, 9:129–135, 1970.

[3] B. Bollobas and E. Cockayne. Graph-theoric parameters concerning domination, independence and irredundance. *Graph Theory*, (3):241–249, 1979.

[4] G. Caporossi and P. Hansen. Variable neighborhood for extremal graphs: 1-the system autographix. *Discrete Mathematics*, 212:29–44, 2000.

[5] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.

[6] S. Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics*, 72:113–118, 1988.

[7] O. Favaron. Stability, domination and irredundance in a graph. *Journal of Graph Theory*, 10:429–438, 1986.

[8] S. Földes and P. L. Hammer. Split graphs. In *Proc. of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 311–315. Congressus Numerantium, 1977.

[9] D. L. Greenwell, R. Hemminger, and J. Klerlein. Forbidden subgraphs. In *Proc. of the Fourth Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 389–394. Congressus Numerantium, 1973.

[10] P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981.

[11] P. Hansen, M. Aouchiche, G. Caporossi, H. Mélot, and D. Stevanovic. What forms do interesting conjectures have in graph theory? *Graphs and Discovery, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, (69):231–252, 2005.

[12] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, 1998.

[13] B. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26:306–324, 1998.

[14] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[15] J. Puech. Irredundance perfection and p6-free graphs. *Journal of Graph Theory*, (29):239–255, 1998.

[16] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory*, 70(B 1):2–44, 1997.

[17] L. Volkmann and V. Zverovich. A proof of a conjecture on irredundance perfect graphs. *Journal of Graph Theory*, (41):292–306, 2002.