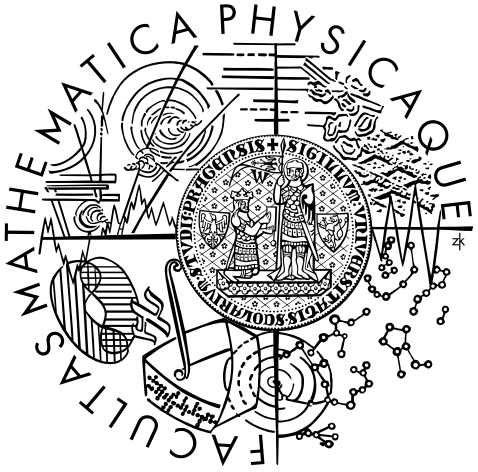
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Eva Ondráčková

Výpočetní složitost v teorii grafů

Katedra Aplikované Matematiky

Vedoucí diplomové práce: Prof. RNDr. Jan Kratochvíl, CSc.
Studijní program: Informatika,
Diskrétní modely a algoritmy

Děkuji prof. Kratochvílovi za cenné vedení a za užitečné rady ohledně podoby práce a psaní matematických textů v angličtině. Dále děkuji doc. Sgallovi za radu o odkazech na literaturu obsahující některé citované výsledky. Jsem vděčná také Vítku Jelínkovi za několik podnětných připomínek a za morální podporu.

Mé díky patří i autorům editoru VRR, ve kterém vznikly všechny obrázky obsažené v této práci.

Prohlašuji, že jsem svou diplomovou práci napsala samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 1. 4. 2006 Eva Ondráčková

Contents

Název práce: Výpočetní složitost v teorii grafů
Autor: Eva Ondráčková
Katedra: Katedra aplikované matematiky
Vedoucí diplomové práce: Prof. RNDr. Jan Kratochvíl, CSc.
e-mail vedoucího: honza@kam.mff.cuni.cz
Abstrakt: Seidelovo přepnutí je grafová operace, která změní hrany vycházející z daného vrcholu tak, aby sousedil s právě těmi vrcholy, které původně nebyly jeho sousedy; zbytek grafu zůstane nezměnen. Dva grafy nazveme ekvivalentní v přepnutí, pokud lze pomocí posloupnosti přepnutí jeden z nich převést na izomorfní tomu druhému. V této práci studujeme výpočetní složitost problému $S(P)$ pro určitou grafovou vlastnost P: je daný graf G ekvivalentní v přepnutí nějakému grafu, který má vlastnost P? Neprve podáváme přehled známých výsledků, vlastností P, pro které je problém $S(P)$ polynomiální, i těch, pro které je NP-úplný. Poté ukážeme NP-úplnost následujícího problému pro každé $c \in (0, 1)$: lze daný graf G přepnout tak, aby obsahoval kliku velikosti alespoň cn, kde n je počet vrcholů grafu G? Zabýváme se také problémem pro pevně zvolený graf H rozhodnout, zda je daný graf G ekvivalentní v přepnutí nějakému H-prostému grafu. Ukážeme, že je-li H izomorfní spáru, tento problém je polynomiální. Dále podáváme charakterizaci grafů, které jsou ekvivalentní v přepnutí nějakému $K_{1,2}$-prostému grafu, pomocí deseti zakázaných indukovaných podgrafů, z nichž každý má pět vrcholů.
Klíčová slova: graf, Seidelovo přepnutí, výpočetní složitost

Title: Computational Complexity in Graph Theory
Author: Eva Ondráčková
Department: Department of Applied Mathematics
Supervisor: Prof. RNDr. Jan Kratochvíl, CSc.
Supervisor's e-mail address: honza@kam.mff.cuni.cz
Abstract: Seidel's switching is a graph operation which makes a given vertex adjacent to precisely those vertices to which it was non-adjacent before, while keeping the rest of the graph unchanged. Two graphs are called switching-equivalent if one can be made isomorphic to the other by a sequence of switches. In this thesis, we study the computational complexity the problem $S(P)$ for a certain graph property P: given a graph G, determine if G is switching-equivalent to a graph having P. First, we give an overview of known results, including both properties P for which $S(P)$ is polynomial, and those for which $S(P)$ is NP-complete. Then we show the NP-completeness of the following problem for each $c \in (0, 1)$: determine if a graph G can be switched to contain a clique of size at least cn, where n is the number of vertices of G. We also study the problem if, for a fixed graph H, a given graph is switching-equivalent to an H-free graph. We show that for H isomorphic to a claw, the problem is polynomial. Further, we give a characterization of graphs switching-equivalent to a $K_{1,2}$-free graph by ten forbidden induced subgraphs, each having five vertices.
Keywords: graph, Seidel's switching, computational complexity

1. Introduction

This thesis studies the computational complexity of several problems related to Seidel's switching of graphs. Seidel's switching is a graph operation which makes a given vertex adjacent to precisely those vertices to which it was non-adjacent before, while keeping the rest of the graph unchanged. Two graphs are called switching-equivalent if one can be made isomorphic to the other by a sequence of switches.

The concept of Seidel's switching was introduced by the Dutch mathematician J. J. Seidel in connection with algebraic structures, such as systems of equiangular lines, strongly regular graphs, or the so-called two-graphs. Seidel laid the foundations of the theory of switching in his work [19–22]. Since then, switching has been studied by many others. In this thesis, we rely mainly on the results of Ehrenfeucht, Hage, Harju, Rozenberg [3–4, 7–9] and Kratochvíl [14–16]. Apart from the algebraic structures, consequences of switching arise in other research fields as well; for example, Seidel's switching plays part in Hayward's polynomial-time algorithm for solving the P_3-structure recognition [10].

In Chapter 2, we introduce the notation and definitions used throughout the thesis. We define Seidel's switching together with several related notions, and present its most useful elementary properties. Then we address the key problem: given a graph G, determine if G is switching-equivalent to a graph possessing a certain property P. This problem is denoted by $S(P)$, and in the following chapters we study the computational complexity of $S(P)$ for various properties P.

Chapter 3 lists several common graph properties P for which deciding $S(P)$ is known to be polynomial. The complexity of recognizing the properties themselves varies; some of them, such as bipartiteness or triangle-freeness, can be recognizable in polynomial time, while the others, like Hamiltonicity, are NP-complete.

In Chapter 4, we focus on NP-complete problems related to switching, mostly on the problems $S(P)$ that are NP-complete. First we give an overview of known results, among them we mention the results for cliques: the problem S("containing a k-clique") is polynomial for each fixed k, but NP-complete if k is a part of the input. In Section 4.5 we extend this by showing that S("containing a clique of size at least cn") is NP-complete even for a fixed $c \in (0, 1)$. Besides the NP-completeness results, in Section 4.6 we show the hardness of approximation of the switching versions of the maximum clique and chromatic number.

In Chapter 5, we examine the complexity of S("being H-free") for several fixed graphs H. Polynomial-time decision algorithms are known for this problem if H has at most three vertices or is isomorphic to a P_4. In Section 5.2 we show that if H is isomorphic to a claw, then the problem is polynomial as well.

Chapter 6 is devoted to characterizations of various switching problems by forbidden induced subgraphs. We give an overview of such known characterizations, and focus on those of S("being H-free") for several graphs H. In Section 6.5 we give such a characterization for H isomorphic to $K_{1,2}$.

2. Basic notions

2.1 Preliminaries

For a set A, let $|A|$ be the cardinality of A. The difference of two sets A and B is denoted by $A \setminus B$, and for the symmetric difference of A and B we write $A \triangle B$. The symbol $\mathbb{N}$ stands for the set of all positive integers $\{1, 2, \dots\}$ and $\mathbb{N}_0$ represents the set of all non-negative integers $\{0, 1, 2, \dots\}$.

A *graph* is a pair $G = (V_G, E_G)$, where V_G is a nonempty set and $E_G \subseteq \binom{V_G}{2}$ is a set of two-element subsets of V_G. We call the elements of V_G *vertices* and the elements of E_G *edges*. Unless defined otherwise, by n we denote the number of vertices of the currently discussed graph. All graphs considered are finite, undirected, and without loops or multiple edges.

Two vertices u, $v \in V_G$ are called *adjacent* if $\{u, v\} \in E_G$, and *non-adjacent* otherwise. The *neighborhood* of a vertex v is the set $N_G(v)$ of all vertices adjacent to v. The *degree* of v is defined by $d_G(v) = |N_G(v)|$. When it is clear which graph is considered, we write only $N(v)$, $d(v)$ etc. We call a vertex *isolated* if it has degree zero. A graph is *k-regular* if all its vertices have degree k.

A graph H is a *subgraph* of a graph G, if $V_H \subseteq V_G$ and $E_H \subseteq \binom{V_H}{2} \cap E_G$. The subgraph H is an *induced subgraph* of G, written $H \leq G$, if $E_H = \binom{V_H}{2} \cap E_G$. For a set $A \subseteq V_G$ we call the graph $(A, \binom{A}{2} \cap E_G)$ the *subgraph of G induced by A* and denote it by $G[A]$. If an isomorphic copy of H is an induced subgraph of G, we shall for simplicity say that G *contains H as an induced subgraph* or just that G *contains* H. We say that a graph G is *H-free* if it does not contain H.

For a vertex $v \in V_G$, the subgraph of G induced by $V \setminus \{v\}$ is denoted by $G - v$. The disjoint union of two graphs G and G' is denoted by $G + G'$.

A graph $G = (V, \binom{V}{2})$ is called a *complete graph* and denoted by K_n. A complete subgraph on k vertices is called a *k-clique*. A cycle of length n is denoted by C_n, a path of length n is denoted by P_n. We say that a graph with n vertices and no edges is *discrete* and denote it by I_n. We call a graph *bipartite* if it can be partitioned into two vertex sets A and B such that both the sets induce discrete subgraphs; and *complete bipartite* if it is bipartite and each vertex of A is adjacent to all vertices of B. By $K_{m,n}$ we denote the complete bipartite graph having $|A| = m$ and $|B| = n$.

2.2 Seidel's switching

Definition 2.1. Let G be a graph. *Seidel's switch of a vertex $v \in V_G$* results in a graph called $S(G, v)$ whose vertex set is the same as of G and the edge set is the symmetric difference of E_G and the full star centered in v, i. e.,

$$V_{S(G,v)} = V_G$$

$$E_{S(G,v)} = E_G \setminus \{xv : x \in V_G,\ xv \in E_G\}) \cup \{xv : x \in V_G,\ x \neq v,\ xv \notin E_G\}.$$

It is easy to observe that the result of a sequence of vertex switches in G depends only on the parity of the number of times each vertex is switched. This allows generalizing switching to vertex subsets of G.

Definition 2.2. Let G be a graph. Then the *Seidel's switch of a vertex subset* $A \subseteq V_G$ is called $S(G, A)$ and

$$S(G, A) = (V_G, E_G \triangle \{xy : x \in A,\ y \in V_G \setminus A\}).$$

The concept of Seidel's switching was introduced by the Dutch mathematician J. J. Seidel in connection with symmetric structures, such as systems of equiangular lines, strongly regular graphs, or the so-called two-graphs. A two-graph, as we mention later, is equivalent to a switching class of graphs.

The following (and many more) basic observations and lemmas about Seidel's switching can be found in [7].

Observation 2.3. *Let G be a graph and A, B two of its vertex subsets. Then*

(1) $S(G, \emptyset) = S(G, V_G) = G$,
(2) $S(G, A) = S(G, V_G \setminus A)$,
(3) $S(S(G, A), A) = G$,
(4) $S(S(G, A), B) = S(S(G, B), A) = S(G, A \triangle B)$,
(5) $S(G, A)[A] = \overline{G[A]}$ *(switching A does not modify edges within A)*,
(6) $S(\overline{G}, A) = \overline{S(G, A)}$.

Lemma 2.4. *Let A and B be vertex subsets of a graph G. Then $S(G, A) = S(G, B)$ if and only if $A = B$ or $A = V_G \setminus B$.*

Proof. If $A = B$ or $A = V_G \setminus B$, then by Observation 2.3 the graphs $S(G, A)$ and $S(G, B)$ are equal.

If $S(G, A) = S(G, B)$, then $S(S(G, A), A) = S(S(G, B), A)$, and by Observation 2.3, $G = S(G, A \triangle B)$. That can be true only if $A \triangle B$ is empty or equal to V_G, which means that $A = B$ or $A = V_G \setminus B$. □

Definition 2.5. We say that two graphs G and H are *switching equivalent* (denoted by $G \sim H$) if there is a set $A \subseteq V_G$ such that $S(G, A)$ is isomorphic to H. The set

$$[G] = \{S(G, A) : A \subseteq V_G\}$$

is called the *switching class* of G.

Note that $\sim$ is an equivalence relation on graphs, and switching classes are the equivalence classes of $\sim$ for graphs on a fixed set of vertices V_G (not considering isomorphism), as shown by Seidel [20–21].

Lemma 2.6. *For a graph G on n vertices, the switching class $[G]$ contains 2^{n-1} graphs.*

Proof. There are 2^n subsets of V_G and by Lemma 2.4 each two different subsets A and B yield a different switch, except for the case when $A = V_G \setminus B$. So we have 2^{n-1} different switches of G. □

Lemma 2.7. *Let G be a graph, $v \in V_G$ and $A \subseteq V_G \setminus \{v\}$. Then there exists a unique graph $H \in [G]$ such that the neighbors of v in H are the vertices of A.*

Proof. In the graph $S(G, N_G(v) \triangle A)$, the neighbors of v are exactly vertices in A. So the graph $S(G, N_G(v) \triangle A)$ has the desired property.

On the other hand, if A is the neighborhood of v in any graph $H = S(G, B)$ for a set B not containing v, then for any vertex $u \in N_G(v) \triangle A$ it holds that $u \in B$, and for any vertex $u \notin N_G(v) \triangle A$ it is true that $u \notin B$. Thus B equals $N_G(v) \triangle A$. Similarly, if $v \in B$ we get that $B = V_G \setminus (N_G(v) \triangle A)$; switching both these sets yields the same graph. □

The following theorem is a variant of Lemma 2.7; it is a consequence of a theorem proved by Zaslavsky [24].

Theorem 2.8. *Let G be a graph. For each tree T on V_G, there exists a unique graph $H \in [G]$ having T as its spanning subgraph.*

Colbourn and Corneil [2] (and independently Kratochvíl et al. [15]) proved that deciding whether two graphs are switching equivalent is an isomorphism-complete problem. However, it is simple to find out if a graph is equal (not only isomorphic) to an element of a given switching class due to the following criterion [20].

Theorem 2.9. (Seidel) *Let $G = (V, E)$, $H = (V, E')$ and $x \in V$. Then $G \in [H]$ if and only if for all triples $T = \{x, y, z\} \subseteq V$, the parity of the number of edges in $G[T]$ equals that of $H[T]$.*

For comparison we introduce the definition of a two-graph.

Definition 2.10. A *two-graph* is a pair (X, V), where X is a set of vertices and V a set of three-element subsets of X, having the property that any four-element subset of X contains an even number of members of V.

For a graph G, there is a corresponding two-graph on the same vertex set whose set of triples consists of the sets of three vertices that induce an odd number of edges of G. It is an immediate consequence of Theorem 2.9 that two graphs are switching-equivalent if and only if they yield the same two-graph. Moreover, there is a one-to-one correspondence between two-graphs and switching classes: every two-graph yields a non-empty switching class [20]. A survey of structural properties of two-graphs can be found in [20–22].

2.3 The complexity of Seidel's switching

Let P be a graph property. We consider the following problem $S(P)$: determine whether a given graph G is switching-equivalent to a graph possessing the property P. In other words, $S(P)$ is the problem of finding out if the switching class $[G]$ contains a graph having P. We want to study the computational complexity of $S(P)$ for various properties P.

Clearly, if recognizing a property P is in NP, then deciding $S(P)$ is also in NP. For an input graph G, it is possible to guess a vertex subset A and then check in

non-deterministic polynomial time if the graph $S(G, A)$ has the property P. We shall deal only with properties which are in NP.

As usual, we say that a property P is *polynomial*, if recognizing P for a graph on n vertices can be done in time polynomial in n, and NP-*complete*, if recognizing P is NP-complete. Additionally, Kratochvíl et al. [15] propose to call a property P

- *switching-polynomial*, if it is polynomial to decide $S(P)$,
- *switching-NP-complete*, if it is NP-complete to decide $S(P)$, and
- *switching-trivial*, if any graph is switching-equivalent to a graph possessing P, or none is.

By Lemma 2.6, there are 2^{n-1} graphs in $[G]$ for a graph G on n vertices. So testing all graphs in $[G]$ separately for possessing the property P would require exponential time. However, there exist properties for which it is polynomial to decide $S(P)$; there are even some for which $S(P)$ is polynomial, but P itself is NP-complete. As observed by Kratochvíl et al. [15] and Ehrenfeucht at al. [4], there is no correlation between the complexity of the problem $S(P)$ and the complexity of the property P itself. For example, Kratochvíl et al. [15] proved that the property "containing a Hamiltonian path" is switching-trivial and the property "containing a Hamiltonian cycle" is switching-polynomial (for more details, see Section 3.3). However, the properties "containing a Hamiltonian path" and "containing a Hamiltonian cycle" themselves are well known to be NP-complete [6].

The property "being a regular graph", on the other hand, is an example of a polynomial but switching-NP-complete property, and "being a k-regular graph" (for a fixed k) is a both polynomial and switching-polynomial one. The property "being 3-colorable" is both NP-complete and switching-NP-complete [16].

In Chapter 3, and partially in Chapters 5 and 6, we give an overview of several switching-polynomial properties. Switching-NP-complete properties are discussed in Chapter 4.

3. Several switching-polynomial properties

In this chapter we summarize known results regarding switching-polynomial properties.

3.1 Graphs of bounded minimum degree

The property "being a k-regular graph" (for a fixed k), mentioned in Section 2.3, is an example of a both polynomial and switching-polynomial property. More generally, it is true that if P is a polynomial property such that the minimum degree of graphs possessing P is bounded by a constant, then P is switching-polynomial, too, as stated by the following theorem [16].

Theorem 3.1. (Kratochvíl) *Let $\mathcal{A}$ be an isomorphism-closed class of graphs such that every graph of $\mathcal{A}$ contains a vertex of degree at most k, for some fixed number k. If $\mathcal{A}$ can be recognized in polynomial time, then it can also be decided in polynomial time if an input graph is switching-equivalent to a graph belonging to $\mathcal{A}$. More precisely, this can be decided in time $\mathcal{O}(n^{k+3}p(n))$, where $p(n)$ is the worst case time complexity of recognizing graphs of $\mathcal{A}$.*

The proof of Theorem 3.1 yields an algorithm to find for a graph G a graph in $[G] \cap \mathcal{A}$. The algorithm tries all $\mathcal{O}(n)$ choices for the vertex v of degree at most k, and for the chosen vertex it tries all of the possible $\mathcal{O}(n^k)$ neighborhoods of v. Such a neighborhood choice defines a unique graph $H \in [G]$ (as observed by Lemma 2.7) and for the graph H it then just checks if $H \in \mathcal{A}$.

It is easy to see that a similar statement can be proved for properties P such that the maximum degree of graphs having P is at least $n - k$ (where n is the number of vertices and k is fixed), i. e. for properties with maximum degree bounded from below.

Theorem 3.1 implies that being a tree, acyclic graph, planar graph, outerplanar graph, graph of bounded genus, graph of bounded tree-width are all switching-polynomial properties.

3.2 Eulerian graphs

We call a graph *even* or *odd*, if all the degrees of its vertices are even or odd, respectively. We say that a graph G is *Eulerian*, if it contains a closed walk visiting each edge of G exactly once. It is well-known that a graph is Eulerian if and only if it is connected and even. Hage et al. [8] found a polynomial-time algorithm to decide the problem S("being an Eulerian graph"). The algorithm is based on the following results.

Theorem 3.2. (Seidel [19]) *Let G be a graph with an odd number of vertices. Then $[G]$ contains a unique even graph.*

Theorem 3.3. (Ehrenfeucht, Hage, Harju, Rozenberg [4]) *Let G be a graph with an even number of vertices. Then either $[G]$ contains no even and no odd graphs, or*

exactly half of its graphs are even while the other half are odd.

Theorem 3.4. (Hage, Harju, Welzl [8]) *Let G be a graph with an even number of vertices and such that $[G]$ contains an even graph. Then $[G]$ contains an Eulerian graph unless $[G]$ contains a complete graph.*

Hence for a graph G with an odd number of vertices, the algorithm finds the unique even graph in $[G]$, and checks whether or not it is connected. The unique even graph can be obtained by switching the set of vertices which have odd degree in G.

For a graph G with an even number of vertices, the algorithm checks if $[G]$ contains an even graph, which is true if and only if G is an even graph or an odd graph; and then it determines if $[G]$ contains a complete graph, which occurs if and only if G is a complete bipartite graph. All this can be done in time $\mathcal{O}(n^2)$.

Apart from the algorithm, Theorem 3.2 is interesting on its own, because it tells us that every graph having an odd number of vertices can be obtained from an even graph by switching. The connection of even graphs to switching has also been studied by Mallows and Sloane [17] who proved that the number e_n of even graphs on n vertices equals the number of switching classes on n vertices. The explicit formula for e_n was given by Robinson [18]:

$$e_n = \sum_{(\sigma)} \frac{2^{\nu(\sigma)-\lambda(\sigma)}}{\prod_i i^{\sigma_i}\sigma_i!},$$

where the sum goes over all ordered n-tuples $\sigma = (\sigma_1, \ldots, \sigma_n)$ such that $n = \sum_i i\sigma_i$ and

$$\nu(\sigma) = \sum_{i<j} \sigma_i\sigma_j \gcd(i,j) + \sum_i i\left(\sigma_{2i} + \sigma_{2i+1} + \binom{\sigma_i}{2}\right),$$

$$\lambda(\sigma) = \sum_i \sigma_i - \operatorname{sgn}\left(\sum_i \sigma_{2i+1}\right),$$

where $\gcd(i,j)$ stands for the greatest common divisor of i and j.

3.3 Hamiltonian and pancyclic graphs

We say that a path (cycle) is a *Hamiltonian path (cycle)* of a graph G if it contains all vertices of G. Deciding if a switching class contains a graph with a Hamiltonian path, and if it contains a graph with a Hamiltonian cycle can be done in polynomial time, as proved by Kratochvíl et al. [15]. An algorithm follows immediately from the following simple characterizations.

Theorem 3.5. *Every graph is switching-equivalent to a graph containing a Hamiltonian path.*[1]

Theorem 3.6. *A graph G is switching-equivalent to a graph containing a Hamiltonian cycle if and only if it is not a complete bipartite graph on an odd number of vertices.*

[1] Theorem 3.5 is also a consequence of Theorem 2.8 – an arbitrary Hamiltonian path can be chosen as the acyclic spanning subgraph T.

These results have been extended to graph pancyclicity by Ehrenfeucht et al. [3]. A graph on n vertices is called *pancyclic* if it has a cycle of length i for all $i = 3, \ldots, n$. Note that pancyclicity is a stronger property than Hamiltonicity, because each pancyclic graph contains a cycle of length n. The characterization is similar to that of Theorem 3.6, except that it excludes complete bipartite graphs on an even number of vertices.

Theorem 3.7. (Ehrenfeucht, Hage, Harju, Rozenberg) *For each graph $G = (V, E)$ on $n \geq 3$ vertices, $[G]$ contains a pancyclic graph if and only if G is not a complete bipartite graph.*

3.4 $K_{1,2}$-free graphs

Kratochvíl et al. [15] found an algorithm which runs in time $\mathcal{O}(n^3)$ and decides if a given graph can be switched not to contain an induced $K_{1,2}$.

The idea of the algorithm is to switch the input graph G so that it makes an arbitrarily chosen vertex v isolated, then remove v and remove every vertex for which there exists an adjacent vertex with the same neighborhood. Then the graph G is switching-equivalent to a $K_{1,2}$-free graph if and only if the reduced graph is a star (a complete bipartite graph $K_{1,m}$ for $m \geq 0$) or a discrete graph.

3.5 Triangle-free graphs and P_3-structures

We call a graph *triangle-free*, if it contains no induced K_3. An algorithm for switching to triangle-free graphs has been found by Hayward [10] in connection with the P_3-structure recognition.

Definition 3.8. A P_k of a graph G is a set of k vertices of G that induces a P_k. A *P_k-structure* of a graph G, written $P_k(G)$, is the set of all P_ks of G. We say that a k-uniform hypergraph H is *realizable* if there exists a graph G such that $P_k(G) = E_H$.

The P_3-structure recognition problem, posed by Chvátal, is the following: Given a 3-uniform graph $H = (V, T)$, find a graph G such that $P_3(G) = T$, or report that no such graph exists. Chvátal also posed an analogous problem regarding P_4-structure recognition. A polynomial-time algorithm for P_4-structure recognition has been found by Hayward et al. [11]; Hayward [10] found an $\mathcal{O}(n^3)$-time algorithm for P_3-structure recognition, using a connection between P_3-structures and two-graphs stated in a theorem equivalent to the following one.

Theorem 3.9. *Let H be a connected 3-uniform hypergraph, and suppose that H is the P_3-structure of a graph G. Then G is I_3-free if H is a two-graph, and if H is not a two-graph, then G is unique and contains at least one of the three graphs on at most five vertices, whose P_3-structure is uniquely realizable.*

As mentioned in Section 2.2 on page 8, a two-graph can be viewed as the set of vertex triples that induce an odd number of edges in a graph. Equivalently, the complementary two-graph is an IP_3-structure of a graph, where the IP_3-structure is a set of triples that induce I_3s or P_3s, three-vertex subgraphs with an even number of edges. So two-graphs and P_3-structures differ mainly in the triples that induce I_3s.

Hayward's algorithm, in case that the input hypergraph H is a two-graph, finds a graph F with H as the corresponding complementary two-graph. Then, using a reduction to 2-SAT, it finds a vertex subset A so that the switch $S(F, A)$ is I_3-free, or decides that no such set A exists.

But if the algorithm finds an I_3-free switch $S(F, A)$ of the graph F, then $\overline{S(F, A)}$ is triangle-free; hence, equivalently, the algorithm finds a triangle-free switch of the graph $\overline{F}$, since $S(\overline{F}, A) = \overline{S(F, A)}$ by Observation 2.3. The algorithm works for any graph F in time $\mathcal{O}(n^3)$.

Independently of this result, Hage et al. [8] also found a $\mathcal{O}(n^3)$-time algorithm for switching to triangle-free graphs that also works by means of a reduction to 2-SAT.

3.6 Bipartite graphs

Hage et al. [8] found a polynomial-time algorithm for determining whether a graph is switching-equivalent to a bipartite graph. This algorithm works by means of a reduction to the 2-SAT problem, too.

Theorem 3.10. (Hage, Harju, Welzl) *Deciding if a switching class contains a bipartite graph or not can be done in time cubic in the number of vertices in the graph.*

As shown by Hage in [7], graphs with a bipartite switch are 4-colorable. The above algorithm, as a by-product, yields a 4-coloring; so it can also be used as an algorithm for finding a 4-coloring for graphs which have a bipartite switch.

It is easy to see that if a switching class contains a complete bipartite graph (or a discrete graph, which is a special case), then it consists only of complete bipartite graphs. It has also been proved in [8] that a graph is (switching-equivalent to) a complete bipartite graph if and only if it does not have an induced K_3 nor $K_2 + K_1$.

4. NP-complete problems for switching classes

In this chapter, we focus on NP-complete problems related to switching. First, we give an overview of known results. Then in Section 4.5 we prove the NP-completeness of the problem Switch-*cn*-Clique, and in Section 4.6 we show the hardness of approximation of Switch-Max-Clique and Switch-Chromatic-Number.

4.1 General results

We shall deal with NP-complete problems for switching classes, mainly with switching-NP-complete properties. It was mentioned in Section 2.3 that the NP-completeness of the problem $S(P)$ for a property P does not imply that the property P itself is NP-complete. However, the following theorem by Kratochvíl et al. [15] says that once we know a switching-NP-complete property, we are sure that there are properties which are both NP-complete and switching-NP-complete.

Theorem 4.1. (Kratochvíl, Nešetřil, Zýka) *Let P be a property such that $S(P)$ is NP-complete. Then S(S(P)) is NP-complete as well.*

There is another general theorem that yields many NP-completeness results for switching classes. We first define several notions that appear in the forthcoming theorem.

Definition 4.2. Let P be a property that is preserved under isomorphisms. We say that P is

- *nontrivial*, if there exists a graph G not having P, and there are arbitrarily large graphs having P;
- *switching-nontrivial*, if P is nontrivial and there exists a switching class $\mathcal{C}$ such that no graph in $\mathcal{C}$ possesses P;
- *hereditary*, if all induced subgraphs of G have the property P whenever G has.

Theorem 4.3. (Ehrenfeucht, Hage, Harju, Rozenberg [4]) *Let P be a switching-nontrivial hereditary property. Then the following problem for instances (G, k) with $k \leq |V_G|$ is NP-hard: does the switching class $[G]$ contain a graph H that has an induced subgraph $H[A]$ with $|A| \geq k$ and such that $H[A]$ has the property P? If recognizing P is in NP, then the corresponding problem is NP-complete.*

For example, "being a complete graph" is a switching-nontrivial hereditary property. As a consequence we get the following corollary.

Corollary 4.4. *For instances (G, k), the problem of deciding if the graph G is switching-equivalent to a graph containing a clique of size at least k is NP-complete.*

If k is fixed (not part of the instances), then the clique problem can be solved by testing all induced subgraphs of size k. The switching class $[G]$ contains a graph

H with a k-clique if and only if at least one induced subgraph of G on k vertices is switching-equivalent to a clique, and that can be determined in polynomial time.

However, in Section 4.5 we extend Corollary 4.4 by proving that the problem of deciding if an input graph G can be switched to contain a clique of size at least cn is also NP-complete, where c is a fixed constant in $(0, 1)$ and n is the number of vertices of G.

The following are some more examples of switching-nontrivial hereditary properties: being a discrete graph, a bipartite graph, a complete bipartite graph, an acyclic graph, a planar graph, a chordal graph; having the chromatic number $\chi(G) \leq l$, where l is a fixed integer. For each of these properties, Theorem 4.3 yields an NP-completeness result. But note that this does not mean that the properties themselves are switching-NP-complete; indeed, some of them have been proved to be switching-polynomial, see Chapter 3.

4.2 The embedding problem

Definition 4.5. We say that a graph H *can be embedded into a graph* G, denoted $H \hookrightarrow G$, if H is isomorphic to a subgraph of G. We write $H \hookrightarrow [G]$, if $H \hookrightarrow S(G, A)$ for some $A \subseteq V_G$.

The embedding problem for graphs is known to be NP-complete [6], and Ehrenfeucht et al. [4] proved that it remains NP-complete for switching classes. The following theorems are consequences of Corollary 4.4.

Theorem 4.6. (Ehrenfeucht, Hage, Harju, Rozenberg) *The embedding problem $H \hookrightarrow [G]$ for switching classes is NP-complete for instances (H, G) of graphs.*

Theorem 4.7. (Ehrenfeucht, Hage, Harju, Rozenberg) *For an instance (G, H, k) of graphs G and H on the same domain D of size n and an integer k with $3 \leq k \leq n-1$, the problem whether there is a set $X \subseteq D$ with $|X| \geq k$ such that $H[X] \in [G[X]]$ is NP-complete.*

They further prove that the problem of embedding a switching class into another, $[H] \hookrightarrow [G]$ for instances (H, G), is NP-complete as well. Note, however, that it is easy to decide if a given graph H is a subgraph of a graph in $[G]$ – it suffices to apply Lemma 2.7.

4.3 Three-colorability of graphs

Given a graph G, we call a function $c : V_G \to C$ (for some set C of colors) a *proper coloring of* G if for all $\{uv\} \in E_G$ it is true that $c(u) \neq c(v)$. A graph G is called *k-colorable* if there is a proper coloring of G that uses only k colors.

The problem of deciding if a given graph is 3-colorable, is well-known to be NP-complete [6]. Ehrenfeucht et al. [4] show that the problem stays NP-complete for switching classes – the problem of deciding whether a switching class $[G]$ contains a 3-colorable graph H, is NP-complete as well.

Kratochvíl [16] further proves that it is NP-complete to decide if an input graph is switching-equivalent to a 3-colorable graph of maximum degree at most 4. Note that

"being a 3-colorable graph of maximum degree at most 4" is a property with bounded minimum degree; but it is NP-complete [6], in contrast with the properties mentioned in Section 3.1.

4.4 Switching to regular graphs

The problem of switching to regular graphs has been studied by Kratochvíl [16]. He proved it to be NP-complete using a reduction to the problem of balancing biregular bipartite graphs, which is related to a variant of hypergraph bicoloring problem.

Here a graph is called *balanced* if its vertices can be colored by two colors (say black and white) so that every vertex has the same number of white and black neighbors. A bipartite graph is (k, r)-*biregular* if every vertex in one bipartition class has degree k and every vertex in the other bipartition class has degree r.

Theorem 4.8. (Kratochvíl) *For every $q \geq 3$, $m \geq 3$ and $1 \leq k \leq m-1$, deciding $(k$-in-$m)$-colorability of q-regular m-uniform hypergraphs is NP-complete.*

Theorem 4.9. (Kratochvíl) *For all $p, q \geq 2$, it is NP-complete to decide if a $(2p, 2q)$-biregular bipartite graph is balanced.*

Theorem 4.10. (Kratochvíl) *Let G be a $(2p, 2q)$-biregular bipartite graph with $n > 2(p+q)$ vertices. If $p \neq q$, then G is switching-equivalent to a regular graph if and only if G is balanced. The resulting graph is then $\frac{n}{2}$-regular.*

Theorem 4.11 follows immediately from Theorem 4.9 and Theorem 4.10.

Theorem 4.11. (Kratochvíl) *Deciding if an input graph is switching-equivalent to a regular graph is NP-complete.*

On the other hand, the problem of deciding if a graph is switching-equivalent to a k-regular graph (for a fixed k) is polynomial, see Section 3.1.

4.5 Searching for a switch with a cn-clique

Definition 4.12. Given a real number $c \in (0, 1)$, we define the problem SWITCH-cn-CLIQUE as follows: decide whether the given graph G is switching-equivalent to a graph containing a clique of size at least cn.

Theorem 4.13. *The problem* SWITCH-cn-CLIQUE *is NP-complete for any $c \in (0, 1)$.*

Proof. We prove the theorem in two steps: first we prove the statement for rational numbers c only; then we extend it to numbers c which are irrational. In the first step, we show the NP-hardness of the problem by reducing SAT to it, whereas in the second step we reduce 3-SAT. Both SAT and 3-SAT are well known to be NP-complete [6]. Finally, we prove that the problem is in NP by finding a polynomial-size certificate.

Suppose that c is rational and equal to $\frac{p}{q}$, where $p, q \in \mathbb{N}$, and $p < q$. We have an instance of SAT: a formula φ in CNF with k clauses and l occurrences of literals, and ask if φ is satisfiable. Without loss of generality we can assume that $k < l$ and $k \geq 2$.

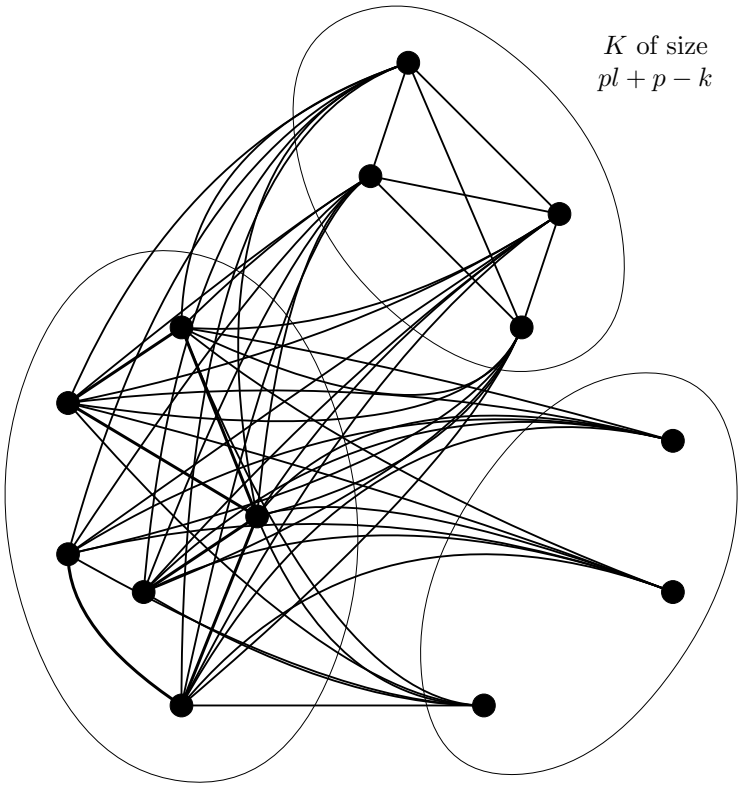


FIG. 4.1. The graph $G_{p,q}(\varphi)$.

Let $G = G_{p,q}(\varphi)$ be a graph constructed in the way illustrated in Figure 4.1. The vertices of G are $V_G = L \cup K \cup Z$, where L, K, Z are pairwise disjoint and

$$|L| = l,$$

$$|K| = pl + p - k,$$

$$|Z| = (q - p - 1)(l + 1) + k + 1.$$

The edges of G are defined as follows:

- K induces a clique and every vertex in K is adjacent to all vertices in L and no vertex in Z.
- Every vertex in Z is adjacent to all vertices in L and nothing more.
- Vertices of L represent occurrences of literals in φ. Two vertices $l_1, l_2 \in L$ are adjacent if and only if
  - l_1 and l_2 occur in different clauses and
  - they are not in the form $l_1 = \neg l_2$ nor $l_2 = \neg l_1$.

Lemma 4.14. *Let j be an integer. The graph $G_{p,q}(\varphi)[L]$ contains a clique on j vertices if and only if the formula φ contains j clauses that are all satisfiable at the same time.*

Proof. If there are j clauses in φ that are simultaneously satisfied by a valuation v, then we can pick from each of them one literal which is true in v; these literals correspond to pairwise adjacent vertices – a clique of size j.

On the other hand, assume that $G[L]$ contains a clique of size j. Then the j corresponding literals cover j clauses and all can be true at the same time. Hence all the j clauses are simultaneously satisfiable. □

Lemma 4.14 immediately gives us the following corollary.

Corollary 4.15. *The formula φ is satisfiable if and only if $G_{p,q}(\varphi)[L]$ contains a clique of size k (where k is the number of clauses in φ).*

Let us now consider cliques of size $pl+p$ in the whole graph – either in the original graph G or in its switches. Note that

$$n = |L \cup K \cup Z| = l + (pl + p - k) + ((q - p - 1)(l + 1) + k + 1) = ql + q$$

$$\frac{pl+p}{n} = \frac{pl+p}{ql+q} = \frac{p(l+1)}{q(l+1)} = \frac{p}{q} = c,$$

therefore cliques of size $pl + p$ are exactly cn-cliques.

Lemma 4.16. *The following statements are equivalent for $G = G_{p,q}(\varphi)$.*

(a) The graph $G[L]$ contains a k-clique.
(b) The graph G contains a $(pl + p)$-clique.
(c) There exists a set $A \subseteq V_G$ such that $S(G, A)$ contains a $(pl + p)$-clique.

Proof. First we prove that (a) implies (b). Any clique in $G[L]$ forms a larger clique together with all vertices of K. So, if $G[L]$ contains a k-clique, then $G[L \cup K]$ contains a clique of size $k + (pl + p - k) = pl + p$.

Obviously (b) implies (c), because if the graph G contains a $(pl + p)$-clique, it suffices to take $A = \emptyset$ for $S(G, A)$ to contain a $(pl + p)$-clique as well.

To prove that (c) implies (a), suppose that there is a set $A \subseteq V_G$ so that $S(G, A)$ contains a $(pl+p)$-clique; let us denote the vertex set of such a clique by C. The set C does not contain more than two vertices of Z, because they are pairwise non-adjacent in G and in $S(G, A)$ they induce a bipartite graph.

From the assumptions $k > l$ and $k \geq 2$ it follows that $l > 2$, and $p \geq 1$, so $pl+p > 2$. Therefore C contains some vertices of L or K. But all vertices of Z are non-adjacent in G and have the same neighborhood in $G[L \cup K]$; surely all vertices in $Z \cap C$ have the same neighborhood in $S(G, A)[C]$ (otherwise C would not induce a clique). But then either $(Z \cap C) \subseteq A$ or $(Z \cap C) \cap A = \emptyset$, so switching A does not affect edges inside $S(G, A)[Z \cap C]$ and any two vertices in $S(G, A)[Z \cap C]$ are non-adjacent. Therefore C contains at most one vertex of Z.

Since $1 + |K| = 1 + (pl + p - k) < pl + p$, the clique C contains at least one vertex of L. But then it cannot contain both vertices of K and Z, because in the graph G they have the same neighborhood in L and there is no edge between K and Z. Also, the set C cannot consist only of vertices of L, because $pl + p > l$. Therefore C contains one of the following:

- $pl + p - 1$ (which is at least k) vertices of L and one vertex of Z
- at least k vertices of L and at least one vertex of K.

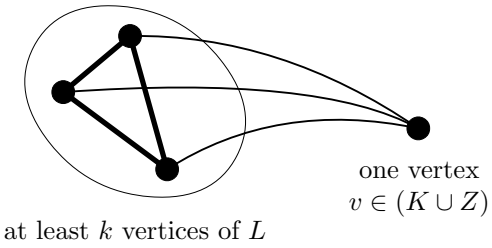


FIG. 4.2. What we have in $S(G, A)$.

In both cases, C contains k vertices of L, and a vertex v of $K \cup Z$. Since C induces a clique in $S(G, A)$, the vertex v is adjacent to all other vertices in C. But in G, by definition, the vertex v is adjacent to all vertices of L, too. So switching A cannot have changed any edge connecting v and the k vertices, which means that either all these $k+1$ vertices are in A or none of them is. But then they induce a $(k+1)$-clique in G as well, and $G[L]$ contains a k-clique, which we wanted to prove. □

Corollary 4.15 and Lemma 4.16 together give us that φ is satisfiable if and only if there exists a set $A \subseteq V_G$ such that $S(G, A)$ contains a $(pl + p)$-clique. But we have already shown that $pl + p = cn$; and clearly a graph contains a clique of size *exactly* cn if and only if it contains a clique of size *at least* cn. That concludes the reduction. The graph $G_{p,q}(\varphi)$ with $q(l+1) = \mathcal{O}(l)$ vertices and $\mathcal{O}(l^2)$ edges can surely be constructed in time polynomial in the size of φ.

Hence the problem SWITCH-cn-CLIQUE is NP-hard for rational constants c. Obviously, it is in NP – a polynomial-size certificate is the vertex subset to induce a clique of the appropriate size. Given such a set A, we pick one vertex $w \in A$ and switch the rest so that w is adjacent to all others. Then by Lemma 2.7, it suffices to check if A really induces a clique; all that can be done in polynomial time. So SWITCH-cn-CLIQUE is NP-complete for any rational constant $c \in (0, 1)$.

To prove the NP-completeness of SWITCH-cn-CLIQUE for irrational numbers c as well, we use a theorem of Arora et al. proved in [1] (and restated in this way in [23] as Lemma 29.10).

Theorem 4.17. (Arora, Lund, Motwani, Sudan, Szegedy) *There exists a polynomial time transformation T from 3-CNF to 3-CNF and a constant $\varepsilon > 0$ such that*

- *If ψ is satisfiable, then $T(\psi)$ is satisfiable.*
- *If ψ is not satisfiable, then at most $1 - \varepsilon$ fraction of the clauses of $T(\psi)$ are simultaneously satisfiable.*

Our aim is to do a reduction from an instance ψ of 3-SAT. We will use the graph $G_{p,q}$, like in the previous part of the proof; this time for the transformed formula $T(\psi)$ and for numbers p and q such that $\frac{p}{q}$ is sufficiently close to the irrational number c. Then we examine the relationship between cn-cliques and $\frac{p}{q}$-cliques in the resulting graph. To show that some suitable numbers p and q exist, we will make use of Lemma 4.18, which is a variant of Dirichlet's Theorem, and Lemma 4.19.

Lemma 4.18. *For any real number $\alpha \in [0, 1]$, any $\varepsilon > 0$ and $r \in \mathbb{R}$ there exists $n \in \mathbb{N}$ such that $n > r$ and $\{n\alpha\} < \varepsilon$ (where $\{n\alpha\}$ stands for the decimal fraction of $n\alpha$).*

Proof. Without loss of generality we can assume that $\varepsilon < \alpha$ and $\alpha \in (0,1)$. We prove by induction that for each $k \in \mathbb{N}_0$ there exists $n_k \in \mathbb{N}$ such that

$$\{n_k\alpha\} \leq \frac{\alpha}{2^k}$$

and $n_{k+1} > n_k$ for all k. Then we take $n = n_k$ for $k \geq \max\{r, \log_2(\frac{\alpha}{\varepsilon})\}$.

We set $n_0 = 1$, because $\{1 \cdot \alpha\} \leq \frac{\alpha}{1}$. Now assume that we already have n_k for some $k \geq 0$ and want to find n_{k+1}. Let $\beta = \{n_k\alpha\}$; we want to get an integer m so that $\{m\beta\} \leq \frac{\beta}{2}$ and $m > 1$. If $\beta = 0$, then clearly the inequality $\{m\beta\} \leq \frac{\beta}{2}$ holds for any integer m, so we can set $m = 2$. Otherwise we consider the number $\lfloor \frac{1}{\beta} \rfloor \beta$. It is clear that $\lfloor \frac{1}{\beta} \rfloor \beta \leq 1$; in case of an equality we have that $\{\lfloor \frac{1}{\beta} \rfloor \beta\} = 0$, while $\lfloor \frac{1}{\beta} \rfloor$ is nonzero. Hence m can be either $\lfloor \frac{1}{\beta} \rfloor$ or any its integral multiple larger than 1.

The remaining case is that $\beta > 0$ and $\lfloor \frac{1}{\beta} \rfloor \beta < 1$. Then $1 < \lceil \frac{1}{\beta} \rceil \beta < \beta + 1$, and after subtracting 1 we get that

$$\left\{ \left\lceil \frac{1}{\beta} \right\rceil \beta \right\} < \beta. \tag{4.1}$$

We want m to be an integer such that $\{m\beta\} \leq \frac{\beta}{2}$. Note that $\lceil \frac{1}{\beta} \rceil > 1$, since $\lfloor \frac{1}{\beta} \rfloor > 0$ for any $\beta \in (0,1)$. So suppose that m cannot be $\lceil \frac{1}{\beta} \rceil$, because $\{\lceil \frac{1}{\beta} \rceil \beta\} > \frac{\beta}{2}$. Then we define

$$\delta = \beta + 1 - \left\lceil \frac{1}{\beta} \right\rceil \beta.$$

The assumption $\{\lceil \frac{1}{\beta} \rceil \beta\} > \frac{\beta}{2}$ together with (4.1) imply that $\delta \in (0, \frac{\beta}{2})$. Similarly like before we obtain the inequalities $\lfloor \frac{\beta}{2\delta} \rfloor \delta \leq \frac{\beta}{2}$ and $\frac{\beta}{2} \leq \lceil \frac{\beta}{2\delta} \rceil \delta$. Moreover, it is surely true that $\lceil \frac{\beta}{2\delta} \rceil \leq (1 + \lfloor \frac{\beta}{2\delta} \rfloor)$; hence

$$\frac{\beta}{2} \leq \left\lceil \frac{\beta}{2\delta} \right\rceil \delta \leq \beta. \tag{4.2}$$

Now we set

$$m = \left(\left\lceil \frac{\beta}{2\delta} \right\rceil \left(\left\lceil \frac{1}{\beta} \right\rceil - 1 \right) + 1 \right).$$

It is clear that such an m is larger than one, and by substituting δ according to its definition, it can be easily verified that

$$m\beta = \left\lceil \frac{\beta}{2\delta} \right\rceil - \left\lceil \frac{\beta}{2\delta} \right\rceil \delta + \beta. \tag{4.3}$$

By plugging the inequalities of (4.2) into (4.3), we obtain

$$\left\lceil \frac{\beta}{2\delta} \right\rceil \leq m\beta \leq \left\lceil \frac{\beta}{2\delta} \right\rceil + \frac{\beta}{2},$$

which immediately gives us that $\{m\beta\} \leq \frac{\beta}{2}$, and that is what we wanted. It now remains to set $n_{k+1} = mn_k$ and verify that $\{n_{k+1}\alpha\} \leq \frac{\alpha}{2^{k+1}}$. Indeed, we have that

$$\{n_{k+1}\alpha\} = \{mn_k\alpha\} = \{m\beta\} < \frac{\beta}{2} = \frac{\{n_k\alpha\}}{2} \leq \frac{\frac{\alpha}{2^k}}{2} = \frac{\alpha}{2^{k+1}},$$

where the last inequality holds by the induction hypothesis. In all considered cases, we chose m to be larger than one, hence $n_{k+1} > n_k$, and we are done. □

Lemma 4.19. *For each irrational $c \in (0,1)$ and $\varepsilon > 0$ there exist p, $q \in \mathbb{N}$ such that $\frac{p}{q} \in (0,1)$ and*

$$c \in \left(\left(1 - \frac{\varepsilon}{4p}\right)\frac{p}{q}, \frac{p}{q}\right).$$

Proof. We shall find an integer p such that the interval $(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c})$ contains another integer q. We want p to satisfy the condition

$$\left\{\frac{p}{c}\right\} < \frac{\varepsilon}{4c}, \tag{4.4}$$

and additionally we request that

$$p > \frac{\varepsilon}{4(1-c)}. \tag{4.5}$$

It is true that $\{\frac{p}{c}\} = \{p\{\frac{1}{c}\}\}$, the number $\{\frac{1}{c}\}$ lies in the interval $[0,1)$, and surely $\frac{\varepsilon}{4c} > 0$; hence Lemma 4.18 for $\alpha = \{\frac{1}{c}\}$, $\varepsilon' = \frac{\varepsilon}{4c}$ and $r = \frac{\varepsilon}{4(1-c)}$ ensures the existence of such a p.

Then we set $q = \lfloor \frac{p}{c} \rfloor$ and verify that it is really an integer in the interval $(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c})$. The number $\frac{p}{c}$ is irrational, so we have $q < \frac{p}{c}$. The fact that $\lfloor \frac{p}{c} \rfloor = \frac{p}{c} - \{p\{\frac{1}{c}\}\}$, and (4.4) together give us the other inequality $q > \frac{p}{c} - \frac{\varepsilon}{4c}$.

Moreover, from (4.5) we obtain

$$\frac{p}{c} - \frac{\varepsilon}{4c} > p,$$

so any integer q in the interval $(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c})$ is larger than p, and thus $\frac{p}{q} \in (0,1)$. Also, by rewriting the inequalities $q < \frac{p}{c}$ and $q > \frac{p}{c} - \frac{\varepsilon}{4c}$ we get the desired inequality

$$\left(1 - \frac{\varepsilon}{4p}\right)\frac{p}{q} < c < \frac{p}{q}.$$

□

Let c be an irrational number in $(0,1)$, let ε be the constant from Theorem 4.17, and p, q the integers given by Lemma 4.19 for ε and c. We take an instance ψ of 3-SAT and construct the graph $G = G_{p,q}(T(\psi))$ in the same way as in the previous part of the proof. Let us again denote the number of clauses of $T(\psi)$ by k. The number of occurrences of literals is $l = 3k$ and n stands for the number of vertices of G.

If ψ is satisfiable, we have again by Corollary 4.15 that $G[L]$ contains a k-clique, and by Lemma 4.16 the graph G contains a $(pl+p)$-clique, which is a $\frac{p}{q}n$-clique. We shall show that if ψ is not satisfiable, then for any set $A \subseteq V_G$ the graph $S(G,A)$ does not contain a clique of size larger than $(1 - \frac{\varepsilon}{4p})\frac{p}{q}n$. We limit ourselves to instances ψ such that $(1-\varepsilon)k > 1$ and $pl + p - \varepsilon k \geq 2$, which we can do without loss of generality. Let us first show the following lemma.

Lemma 4.20. *Let ψ be a formula such that $(1-\varepsilon)k > 1$ and $pl + p - \varepsilon k \geq 2$. If ψ is not satisfiable, then for any set $A \subseteq V_G$ the graph $S(G, A)$ does not contain a clique of size larger than $pl + p - \varepsilon k$.*

Proof. Suppose that for some $A \subseteq V_G$ we have a clique on a vertex set C in $S(G, A)$ and the size of the clique is larger than $pl + p - \varepsilon k$. Then (similarly as in the proof of Lemma 4.16) we get that the set C does not contain more than two vertices of Z, because they are pairwise non-adjacent in G and in $S(G, A)$ they induce a bipartite graph.

Since $|C|$ is more than two, then C contains some vertices of L or K. But all vertices of Z are non-adjacent in G and have the same neighborhood in $G[L \cup K]$; surely all vertices in $Z \cap C$ have the same neighborhood in $S(G, A)[C]$ (otherwise C would not be a clique). But then either $(Z \cap C) \subseteq A$ or $(Z \cap C) \cap A = \emptyset$, so switching A does not affect edges inside $S(G, A)[Z \cap C]$, and any two vertices in $S(G, A)[Z \cap C]$ are non-adjacent. Therefore C contains at most one vertex of Z.

The set C contains at least one vertex of L, because

$$1 + |K| = 1 + (pl + p - k) < (1-\varepsilon)k + pl + p - k = pl + p - \varepsilon k.$$

But then C cannot contain both vertices of K and Z, because in G they have the same neighborhood in L and there is no edge between K and Z.

By Lemma 4.14, every clique in L corresponds to $|C \cap L|$ clauses which are simultaneously satisfiable. Hence by Theorem 4.17, the maximum clique size in L is $(1-\varepsilon)k$, which is not enough for C, since

$$(1-\varepsilon)k < (1-\varepsilon)k + pl + p - k = pl + p - \varepsilon k,$$

hence C cannot consist only of vertices of L. Therefore C consists of one of the following:

- more than $pl + p - \varepsilon k - 1$ (which is larger than $(1-\varepsilon)k$) vertices of L, and one vertex of Z
- more than $(1-\varepsilon)k$ vertices of L, and $pl + p - k$ vertices of K.

In both cases, C contains more than $(1-\varepsilon)k$ vertices of L, and a vertex v from K or Z. Since C induces a clique in $S(G, A)$, the vertex v is adjacent to all other vertices in C. But in G, by definition, v is adjacent to all vertices of L, too. So switching A cannot have changed any edge connecting v and the other vertices, which means that either all the vertices are in A or none of them is. But then they induce a clique of size larger than $(1-\varepsilon)k$ in $G[L]$ as well. As we have already shown, the maximum clique size in $G[L]$ is $(1-\varepsilon)k$, which is a contradiction. □

By Lemma 4.20, if ψ is not satisfiable, then the maximum clique size in $S(G, A)$ for any A is $pl + p - \varepsilon k$. But

$$\frac{\varepsilon k}{n} = \frac{\varepsilon k}{q(l+1)} = \frac{\varepsilon k}{q(3k+1)} \geq \frac{\varepsilon}{4q} = \frac{\varepsilon}{4p} \cdot \frac{p}{q},$$

so the maximum clique size divided by n is

$$\frac{pl + p - \varepsilon k}{n} = \frac{p(l+1)}{q(l+1)} - \frac{\varepsilon k}{q(l+1)} \leq \frac{p}{q} - \frac{\varepsilon}{4p} \cdot \frac{p}{q} = \left(1 - \frac{\varepsilon}{4p}\right) \frac{p}{q}.$$

We have chosen the numbers p, q so that

$$c \in \left(\left(1 - \frac{\varepsilon}{4p}\right) \frac{p}{q}, \frac{p}{q} \right),$$

hence the maximum clique ratio matches the lower bound of the interval containing c. To sum it all up, we have shown that

- if ψ is satisfiable, then there exists an $A \subseteq V_G$ such that $S(G, A)$ contains a clique of size $\frac{p}{q}n$, which is at least cn,
- if ψ is not satisfiable, then for no set $A \subseteq V_G$ the graph $S(G, A)$ contains a clique of size more than $(1 - \frac{\varepsilon}{4p})\frac{p}{q}n$, especially of size at least cn.

Hence ψ is satisfiable if and only if G can be switched to contain a clique of size at least cn. The graph $G_{p,q}(T(\psi))$ with $q(l+1) = \mathcal{O}(l)$ vertices and $\mathcal{O}(l^2)$ edges can be constructed in polynomial time. That concludes the polynomial-time reduction of 3-SAT to Switch-cn-Clique for an irrational constant c, and also the proof that the problem is NP-hard.

Again, Switch-cn-Clique is in NP – a polynomial-size certificate is the vertex subset to induce a clique of the appropriate size. Given such a set A, we pick one vertex $w \in A$ and switch the rest so that w is adjacent to all others. Then by Lemma 2.7 it suffices to check if A really induces a clique; all that can be done in polynomial time.

So the problem Switch-cn-Clique is NP-complete for all constants $c \in (0, 1)$, which completes the proof. □

4.6 Hardness of approximation for switching problems

In addition to the NP-completeness results, we also get the hardness of approximation of several switching problems. We show that for the problems discussed below, allowing switching of the input graph does not make any significant difference in the size of the approximated quantity. So the non-approximability results are analogous to those for the original problems without switching.

We consider the following problems (all having a graph G as the instance).

Max-Clique	What is the size of a largest clique in G?
Switch-Max-Clique	What is the size of a largest clique in $S(G, A)$ over all $A \subseteq V_G$?
Chromatic-Number	What is the minimum number of colors needed to color G properly?
Switch-Chromatic-Number	What is the minimum number of colors needed to color properly at least one of $S(G, A)$ for $A \subseteq V_G$?

Claim 4.21. *Let $\alpha \geq 1$. If there is a polynomial-time α-approximation algorithm for* Switch-Max-Clique*, then there also exists a polynomial-time 2α-approximation algorithm for* Max-Clique*.*

Proof. Suppose that $\mathcal{A}$ is a polynomial-time α-approximation algorithm for Switch-Max-Clique. Let G be a graph, M the size of a largest clique in G, let M_S be the

size of a largest clique in the graphs $S(G, A)$ over all vertex subsets A, and $k = |\mathcal{A}(G)|$. Since $\mathcal{A}$ is an α-approximation algorithm, we have that

$$k \geq \frac{1}{\alpha} M_S.$$

Let A be a set for which $S(G, A)$ contains the clique $C = \mathcal{A}(G)$ of size k. Note that if we know the vertices of C, we can in polynomial time find such a set A according to Lemma 2.7; for any vertex $v \in C$, we can set $A = C \setminus N_G(v)$.

Clearly, one of A and $V_G \setminus A$, say A, contains at least half of the vertices of C. By Observation 2.3, $S(G, A)[A] = G[A]$; hence $C' = A \cap C$ is a clique in G of size at least $\frac{1}{2}k$. Surely $M_S \geq M$, because G also counts as its switch $S(G, \emptyset)$. Therefore

$$|C'| \geq \frac{1}{2}k \geq \frac{1}{2\alpha} M_S \geq \frac{1}{2\alpha} M,$$

so C' is a clique in G of size within the approximation factor, and we are done. □

Similarly we prove an analogous result regarding the approximation of the chromatic number. Note that Switch-Max-Clique and Max-Clique are maximization problems, while Switch-Chromatic-Number and Chromatic-Number are minimization ones, so the inequalities are reversed. The proof is similar to that of Lemma 4.22 (see Lemma 3.30 in [7]).

Lemma 4.22. (Hage, Harju) *Let G be a graph with $\chi(G) = k$. Then for all switches $G' = S(G, A)$ of G, $k/2 \leq \chi(G') \leq 2k$.*

Claim 4.23. *Let $\alpha \geq 1$. If there is a polynomial-time α-approximation algorithm for* Switch-Chromatic-Number*, then there is also a polynomial-time 2α-approximation algorithm for* Chromatic-Number*.*

Proof. Suppose that there is a polynomial-time α-approximation algorithm $\mathcal{A}$ for Switch-Chromatic-Number. Let G be a graph, $\chi = \chi(G)$, let χ_S be the minimum chromatic number of $S(G, A)$ over all vertex subsets A, let c be the coloring $\mathcal{A}(G)$, and k the number of colors used in c. Since $\mathcal{A}$ is an α-approximation algorithm, we have that

$$k \leq \alpha \chi_S.$$

Let A be a set for which $S(G, A)$ can be properly colored by c. Such a set A can be constructed from c in the following way: to each vertex v we assign a boolean variable x_v, and for each vertex pair u, v such that $c(u) = c(v)$, we form a condition

- $(x_u \vee x_v)$ & $(\neg x_u \vee \neg x_v)$ if u and v are adjacent in G or
- $(x_u \vee \neg x_v)$ & $(\neg x_u \vee x_v)$ if they are not.

The conjunction of all these conditions is an instance of 2-SAT, and any its satisfying valuation corresponds to a set $A = \{v \in V(G) : x_v = 1\}$ with the property that $S(G, A)$ is properly colored by c. Since c is a proper coloring, we know that such a valuation exists; moreover, it can be found in time polynomial in the number of vertices, and the 2-SAT instance can be constructed in polynomial time, too.

Suppose that $c_1, \ldots, c_k$ are the colors used by c. We define k other colors $c'_1, \ldots, c'_k$, a coloring

$$c'(v) := \begin{cases} c(v) & \text{if } v \in A \\ (c(v))' & \text{if } v \notin A, \end{cases}$$

and observe that c' is a proper coloring of G.

Surely $\chi_S \leq \chi$, because G also counts as its switch $S(G, \emptyset)$. We have that c' uses k' colors, where

$$k' = 2k \leq 2\alpha\chi_S \leq 2\alpha\chi,$$

so the number of colors used by c' is within the approximation factor. □

Let us combine Claim 4.21 and Claim 4.23 with known results regarding the hardness of approximation of Max-Clique and Chromatic-Number. Håstad [12] proved that for any $\varepsilon > 0$, there is no polynomial-time $n^{1-\varepsilon}$-approximation algorithm for Max-Clique, unless NP = co-RP. A similar result has been proved by Feige and Kilian [5] for Chromatic-Number. They show that for any $\varepsilon > 0$, there is no polynomial-time $n^{1-\varepsilon}$-approximation algorithm for Chromatic-Number, unless NP = co-RP. We conclude the following corollary.

Corollary 4.24. *Let $\varepsilon > 0$. Unless* NP = co-RP*, there is no polynomial-time approximation algorithm for* Switch-Max-Clique *nor* Switch-Chromatic-Number *within the factor $n^{1-\varepsilon}$.*

Proof. Suppose that for a certain $\varepsilon > 0$, there exists a polynomial-time $n^{1-\varepsilon}$-approximation algorithm for one of the problems, say Switch-Max-Clique (the case of Switch-Chromatic-Number is analogous). Then by Claim 4.21, there is a polynomial-time $2n^{1-\varepsilon}$-approximation algorithm for Max-Clique. Clearly, if n is large enough, then $2n^{1-\varepsilon} < n^{1-\varepsilon/2}$; hence there is also a polynomial-time $n^{1-\varepsilon/2}$-approximation algorithm for Max-Clique, which is a contradiction. □

5. Switching to H-free graphs

In this chapter, we examine the complexity of the problem S("being H-free") for several fixed graphs H. Polynomial-time decision algorithms are known for this problem if H has at most three vertices or is isomorphic to a P_4. In Section 5.2 we show that if H is isomorphic to a claw, then the problem is polynomial as well.

5.1 General observations

Let H be a graph. For H fixed, it is easy to decide if a given graph G can be switched to contain H as an induced subgraph. It suffices to try all induced subgraphs of G on $|V_H|$ vertices and test them according to Lemma 2.7 for being switching-equivalent to H. It is clear that if an induced subgraph of G can be switched to contain H, then the same is true for the whole graph G. This leads to the following observation.

Observation 5.1. *For each graph H the problem S("containing H") can be solved in time polynomial in the size of the input graph.*

In this chapter, we examine the opposite problem: is the graph G switching-equivalent to a graph not containing H as an induced subgraph? For this problem there is no such simple general observation. However, all known results for several particular graphs H yield polynomial-time algorithms. No graph H is known for which S("being H-free") is NP-complete, and it is not clear if such a graph exists.

By Observation 2.3, for a set $A \subseteq V_G$ it is true that $S(\overline{G}, A) = \overline{S(G, A)}$. So the graph $S(G, A)$ is H-free if and only if the graph $S(\overline{G}, A)$ is $\overline{H}$-free. That gives us another observation.

Observation 5.2. *If an algorithm $\mathcal{A}$ decides whether an input graph G is switching-equivalent to an H-free graph, then the algorithm $\mathcal{A}$ for the input graph $\overline{G}$ also decides whether G is switching-equivalent to an $\overline{H}$-free graph.*

Hence, in the following text, we shall identify the algorithm that solves the problem for a graph H with the one that solves it for $\overline{H}$.

For K_2 (and I_2), the problem is simple: K_2-free graphs are discrete, and graphs switching-equivalent to discrete graphs are exactly complete bipartite graphs, as mentioned in Section 3.6 on page 13. Switching to triangle-free (and I_3-free) graphs is discussed in Section 3.5 on page 12, and for switching to $K_{1,2}$-free (and $(K_1 + K_2)$-free) graphs, see Section 3.4 on page 12. That covers all graphs H on at most three vertices.

As for graphs on four vertices, a polynomial-time algorithm for switching to P_4-free graphs follows from the characterization of such graphs given in Section 6.2 on page 30. In Section 5.2, we show that the problem of switching to $K_{1,3}$-free (and $(K_1 + K_3)$-free) graphs can be decided in polynomial time, too.

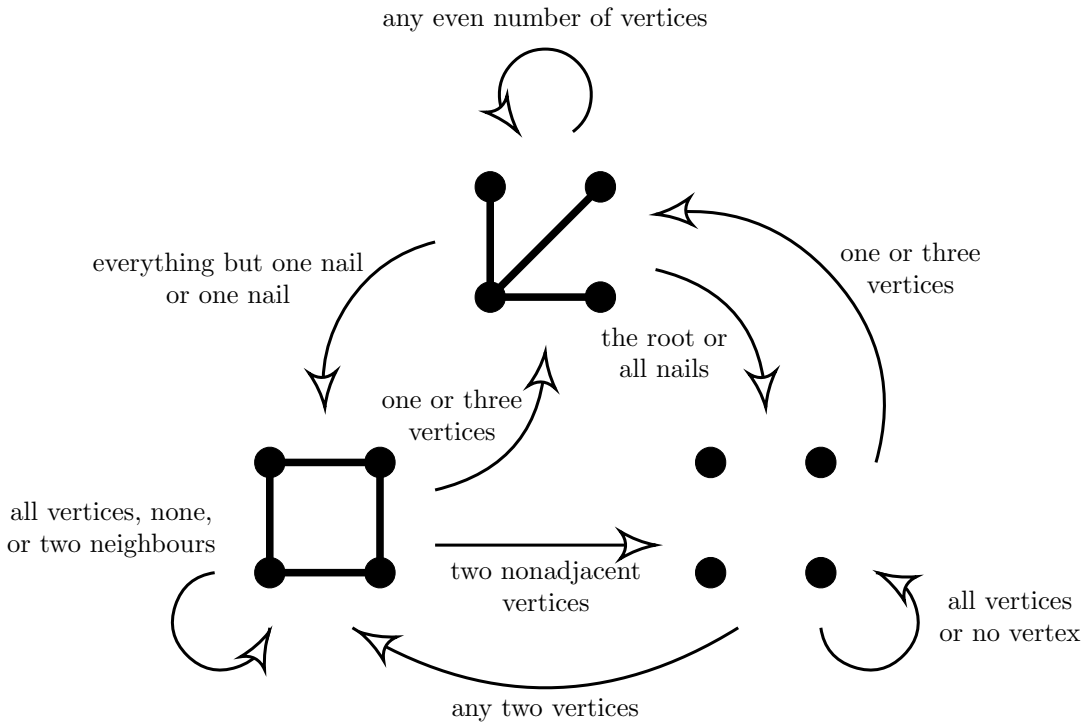


FIG. 5.3. Dangerous graphs and switching between them.

5.2 Switching to a claw-free graph in polynomial time

In this section, we show that it is polynomial to decide if a given graph can be switched to a claw-free graph.

Theorem 5.3. *Given a graph G, we can in polynomial time find a set $A \subseteq V_G$ such that $S(G, A)$ is claw-free, or find out that no such A exists.*

Definition 5.4. We call a graph on four vertices a *dangerous graph* if it is switching-equivalent to a claw. We say that the vertex of degree three in a claw is the *root* of the claw; vertices of degree one are *nails* of the claw.

Lemma 5.5. *All the dangerous graphs (up to isomorphism) are the claw $K_{1,3}$ itself, the four-cycle C_4 and four isolated vertices I_4. By switching an even number of vertices in a dangerous graph, we obtain*

- *a claw from a claw*
- *a non-claw from a non-claw.*

By switching an odd number of vertices in a dangerous graph, we obtain

- *a claw from a non-claw*
- *a non-claw from a claw.*

Proof. All possible switches between the graphs $K_{1,3}$, C_4, and I_4 are demonstrated in Figure 5.3. A case analysis of the switches follows.

Switching four or zero vertices does not modify the graph, therefore it does not change the property of being a claw; it remains to consider switching one, two or three vertices. Switching any two vertices of a claw creates a claw. Switching the root or all nails of a claw yields an I_4, and switching one nail or everything but one nail yields a C_4. Thus switching any vertex subset of a claw results in one of $K_{1,3}$, C_4, and I_4, and those are really all the dangerous graphs (up to isomorphism). Moreover, we get a claw from a claw if and only if we switch an even number of its vertices.

Switching one or three vertices in a C_4 gives us a claw. By switching two adjacent vertices we get a C_4, and by switching two non-adjacent vertices we obtain an I_4. Thus we get a claw from a C_4 if and only if we switch an odd number of vertices.

Switching one or three vertices in an I_4 yields a claw, whereas switching two yields a C_4. Again, a claw arises from an I_4 if and only if an odd number of vertices are switched. □

Corollary 5.6. *Let G be a graph and $A \subseteq V_G$. Then $S(G, A)$ is claw-free if and only if for every dangerous induced subgraph H of G the following is true:*

- *$|V(H) \cap A|$ is odd if H is a claw,*
- *$|V(H) \cap A|$ is even if H is a not claw.*

Proof. By Lemma 5.5, switching such an A creates a non-claw from every (dangerous) claw and a non-claw from every dangerous non-claw. But a claw can arise only from a dangerous graph; therefore switching A destroys all claws and creates no new one.

Conversely, if a set A yields a claw-free switch, then by Lemma 5.5 it contains an odd number of vertices out of every dangerous claw, and an even number of vertices out of every dangerous non-claw. □

Proof. (of Theorem 5.3) We show that vertex subsets with the desired properties can be described by a system of linear equations over $GF(2)$ with $\mathcal{O}(n)$ variables and $\mathcal{O}(n^4)$ equations. Solutions of such an equation system can be computed, for example, by the Gaussian elimination in time $\mathcal{O}(n^6)$. In particular, it can be decided in time $\mathcal{O}(n^6)$ whether a solution of such a system exists. The equations, too, can be constructed in polynomial time.

We compute in $GF(2)$. To every vertex $v \in V_G$ we assign a variable x_v, and for every dangerous subgraph H on vertices v_i, v_j, v_k, v_l, we form an equation

$$x_{v_i} + x_{v_j} + x_{v_k} + x_{v_l} = 1$$

if H is a claw or

$$x_{v_i} + x_{v_j} + x_{v_k} + x_{v_l} = 0$$

if H is not a claw. Clearly, we get at most $\binom{n}{4} = \mathcal{O}(n^4)$ equations in this way.

Every assignment of values to the variables yields a vertex subset

$$A = \{v \in V_G : x_v = 1\}.$$

The equations express parity requirements for the size of the intersection of A and dangerous subgraphs. Then, according to Corollary 5.6, the solutions of this system correspond to all vertex subsets A such that $S(G, A)$ is claw-free. □

6. Characterizations by forbidden induced subgraphs

In this chapter, we study characterizations of various switching problems by a list of forbidden induced subgraphs. First, we summarize known results, and in Section 6.5 we give such a characterization for S("being $K_{1,2}$-free").

6.1 Remarks

In Section 2.3 we mention several switching-polynomial properties. There are yet some more properties that are known to be switching-polynomial – these are properties P such that $S(P)$ can be characterized by a finite set of forbidden induced subgraphs. It is clear that such a characterization yields a polynomial-time decision algorithm.

A simple example of such a characterization is the one of S("being discrete") or S("being complete bipartite") mentioned in Section 3.6. A graph can be switched to a discrete graph if and only if it does not contain an induced K_3 nor $K_1 + K_2$. Equivalently, such graphs can be switched to a complete bipartite graph, as shown by Hage et al. [8].

Now suppose that P is a hereditary property (recall that a property P is *hereditary* if all induced subgraphs of a graph G have the property P whenever G has). If there exists an induced subgraph of a graph G that cannot be switched to posses P, then neither G can. So $S(P)$ is hereditary as well, and the badness of a graph is always caused by a bad induced subgraph. A set of forbidden induced subgraphs for $S(P)$ can be obtained as the set of all bad graphs, or the set of all minimal bad graphs. But such a set may be infinite. That is also the case of the S("being acyclic") characterization (see Section 6.3). The most useful cases are those when the set is finite and reasonably small.

Conversely, if a property is shown to be equivalent to not containing any bad induced subgraph, which is hereditary, then the property itself is hereditary.

In the following sections, several known characterizations by forbidden induced subgraphs are listed. Some of them, instead of $S(P)$, characterize switching classes whose all graphs have a certain property.

6.1.1 Remarks on switching to H-free graphs

In Chapter 5, we examine the complexity of S("being H-free") for fixed graphs H. We give an overview of several graphs H for which the property is switching-polynomial; in fact, all known results for particular graphs H yield polynomial-time algorithms. No graph H is known for which S("being H-free") is NP-complete.

For some of those graphs H, a characterization of S("being H-free") by a finite list of forbidden subgraphs has been found. These are the graphs K_2, I_2 (due to Hage et al. [8]), and P_4 (due to Hertz [13]). In Section 6.5 we extend them by $K_{1,2}$.

Again, there is a relation between results for H and for $\overline{H}$. It has been mentioned in Chapter 5 that for a graph G the graph $S(G, A)$ is H-free if and only if the graph

$S(\overline{G}, A)$ is $\overline{H}$-free (by Observation 2.3). That gives us the following observation.

Observation 6.1. *Let n be an integer, and H, F_1, ... , F_n graphs. The following statements are equivalent:*

(1) For every graph G, the switching class $[G]$ contains an H-free graph if and only if $F_i \not\leq G$ for any $i = 1, \ldots, n$.
(2) For every graph G, the switching class $[G]$ contains an $\overline{H}$-free graph if and only if $\overline{F_i} \not\leq G$ for any $i = 1, \ldots, n$.

Hence a finite list of forbidden subgraphs for H also yields one of the same size for $\overline{H}$; and the result for $K_{1,2}$ also gives a list for $K_2 + K_1$. In case of the triangle K_3 and the claw $K_{1,3}$, the corresponding polynomial-time algorithms exist (see Section 3.5 on page 12 and Section 5.2 on page 27), but the lists of forbidden induced subgraphs are not known to be finite. Moreover, it seems that there is not a reasonably short list for either of them. According to the computational results, both lists soon grow to enormous size, which we find interesting in view of the fact that the list for $K_{1,2}$ has ten graphs only, and the graphs K_3 and $K_{1,2}$ differ just in one edge.

On the other hand, no graph H is known such that the appropriate list of forbidden induced subgraphs is infinite; that allows one to believe that the list is finite for every H.

6.2 Perfect graphs and P_4-free graphs

A graph is called *perfect* if for every induced subgraph G, the chromatic number $\chi(G)$ equals the clique number $\omega(G)$. A set of forbidden induced subgraphs for graphs whose all switches are perfect was found by Hertz [13]. The forbidden graphs are shown in Figure 6.4.

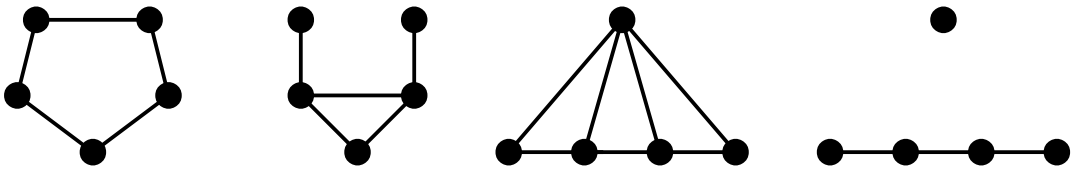


FIG. 6.4. Forbidden induced subgraphs for perfectness of a switching class.

Theorem 6.2. (Hertz) *For a switching class $\mathcal{S}$, the following statements are equivalent:*

(1) $\mathcal{S}$ contains only perfect graphs.
(2) Every graph in $\mathcal{S}$ is C_5-free, bull-free, gem-free and anti-gem-free.
(3) Some graph in $\mathcal{S}$ is P_4-free.

Theorem 6.2 gives a characterization of graphs switching-equivalent to a P_4-free graph. Since C_5, bull, gem and anti-gem together are (up to isomorphism) a switching class, then a graph G is C_5-free, bull-free, gem-free and anti-gem-free if and only if every graph in $[G]$ is C_5-free, bull-free, gem-free and anti-gem-free, too. Hence if G does not contain any of these four graphs, then, according to Theorem 6.2, $\mathcal{S} = [G]$

contains a P_4-free graph. And vice versa, if a graph in $\mathcal{S} = [G]$ is P_4-free, then G does not contain any of the forbidden induced subgraphs.

Therefore switching classes containing a P_4-free graph can be characterized by four forbidden induced subgraphs on five vertices.

6.3 Acyclic graphs

Switching classes containing acyclic graphs have been examined by Hage and Harju in [9]. They have found a characterization of such classes by a list of forbidden induced subgraphs. Apart from the cycles C_n for $n \geq 7$, there are only finitely many graphs in the list, each having at most nine vertices. They prove that there are 905 forbidden subgraphs partitioned into 27 switching classes (up to isomorphism and excluding switches of cycles C_n). A computer program was employed to obtain this result.

The infinite list of forbidden subgraphs does not immediately give us an efficient recognition algorithm. However, it is polynomial to decide S("being acyclic"), since "being acyclic" is a property of bounded minimum degree (see Section 3.1).

6.4 Perfect codes

Definition 6.3. Given a graph G, a set $C \subseteq V_G$ is called a *t-perfect code* in G if and only if the sets $S_t(u) = \{v : v \in V_G \ \& \ d(u, v) \leq t\}$ for all $u \in C$ form a partition of V_G, i. e., if for every $v \in V_G$ there is exactly one $u \in C$ such that $d(u, v) \leq t$.

Definition 6.4. We say that a switching class $\mathcal{S}$ is *t-codeperfect* if each $H \in \mathcal{S}$ contains a t-perfect code.

Kratochvíl [14] shows that a switching class $[G]$ is 1-codeperfect if and only if G contains none of the seven graphs depicted in Figure 6.5 as an induced subgraph.

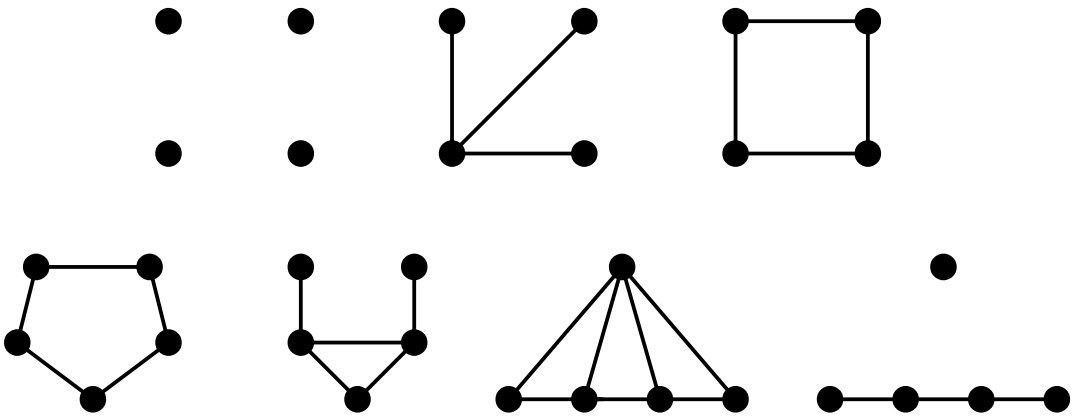


FIG. 6.5. Forbidden induced subgraphs for 1-codeperfect switching classes.

Note that a 1-perfect code in a graph G is also a perfect dominating set for G. (A set $C \subseteq V_G$ is *dominating* if for all $v \in V_G$ either $v \in C$ or v is adjacent to a vertex in C, and a dominating set C is *perfect* if each vertex $v \in V_G \setminus C$ is adjacent to exactly one vertex of C.) Hence the forbidden induced subgraphs in Figure 6.5 also characterize switching classes containing only graphs with perfect dominating sets.

6.5 Switching classes containing a $K_{1,2}$-free graph

In this section, we give a characterization of switching classes containing a $K_{1,2}$-free graph by ten forbidden induced subgraphs, each having five vertices. This characterization, however, does not bring any improvement for the algorithmic recognition of switching classes containing a $K_{1,2}$-free graph; checking all induced subgraphs on five vertices requires time $\Omega(n^5)$, whereas the already known algorithm of Kratochvíl et al. [15] runs in time $\mathcal{O}(n^3)$. This algorithm is described in section 2.3 on page 8.

The characterization is interesting in view of the question whether for all graphs H the list of forbidden induced subgraphs for S("being H-free") is finite. We show that for $H = K_{1,2}$, it is.

Theorem 6.5. *A graph G is switching-equivalent to a $K_{1,2}$-free graph if and only if G does not contain any of $G_1, \ldots, G_{10}$ as an induced subgraph.*

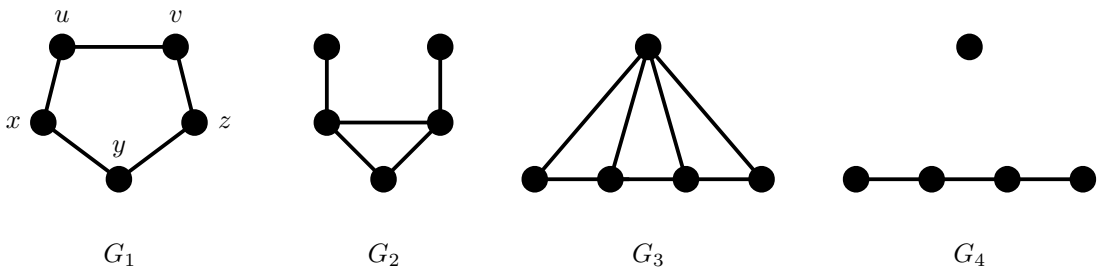


FIG. 6.6. Forbidden graphs switching-equivalent to G_1.

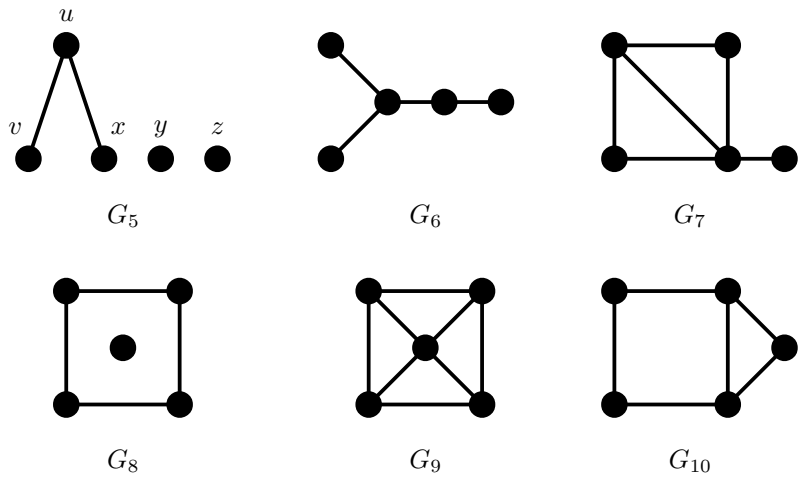


FIG. 6.7. Forbidden graphs switching-equivalent to G_5.

Lemma 6.6. *The graphs G_1, G_2, G_3, G_4 are (up to isomorphism) all the graphs switching equivalent to G_1. None of them can be switched to a $K_{1,2}$-free graph.*

Proof. Since being switching-equivalent is an equivalence relation, it suffices to examine all the possible switches of G_1. Without loss of generality the set A of switched vertices is of size at most two (otherwise we switch $A' = V_G \setminus A$ instead of A, which

gives us the same resulting graph). By switching one vertex in G_1 we obtain a G_2. By switching two adjacent vertices in G_1 we get a G_3, and switching two non-adjacent ones gives us a G_4.

It remains to show that none of the graphs G_1, G_2, G_3, G_4 can be switched to a $K_{1,2}$-free graph; in other words, the switching class of G_1 does not contain a $K_{1,2}$-free graph. Indeed, none of G_1, G_2, G_3, G_4 is $K_{1,2}$-free, as can be seen in Figure 6.6. □

Lemma 6.7. *The graphs $G_5, \ldots, G_{10}$ are (up to isomorphism) all the graphs switching equivalent to G_5. None of them can be switched to a $K_{1,2}$-free graph.*

Proof. Similarly as in the proof of Lemma 6.6, we examine all the possible switches of G_5 and without loss of generality we switch at most two vertices. By switching u we get a G_5. By switching one of v, x we get a G_6. Switching one of y, z produces a G_7. Switching u and one of v, x yields a G_7. By switching u and one of y, z we get a G_6. By switching v and x we get a G_8. Switching y and z produces a G_9 and, finally, switching one of v, x and one of y and z creates a G_{10}.

It remains to show that none of the graphs $G_5, \ldots, G_{10}$ can be switched to a $K_{1,2}$-free graph. Again, as observed in Figure 6.7, none of the elements of this switching class is $K_{1,2}$-free. □

Lemma 6.8. *Let G' be a graph not containing any of $G_1, \ldots, G_{10}$ as an induced subgraph. Then G' is switching-equivalent to a $K_{1,2}$-free graph.*

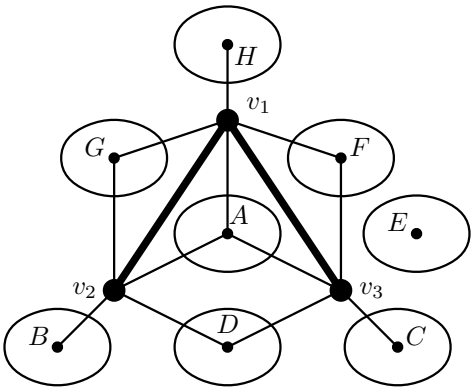
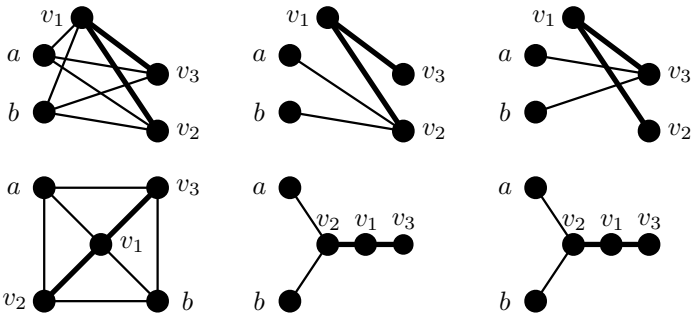


FIG. 6.8. Eight possibilities for a vertex's neighborhood in a $K_{1,2}$.

Proof. If G' does not contain any $K_{1,2}$, we are done. Otherwise we choose a $K_{1,2}$ in G', denote its vertex of degree two by v_1 and the other ones v_2 and v_3. We divide the remaining vertices of G' into eight groups according to their neighborhood in v_1, v_2, v_3, as marked in Figure 6.8. Our aim now is to show that the groups A, B, C, E, F, G are cliques, whereas D and H are disjoint unions of cliques.

In Figure 6.9, the upper left graph represents one of the forbidden situations: it contains the fixed $K_{1,2}$ with its vertices denoted by v_1, v_2, v_3, and two vertices $a, b \in A$ along with all the edges connecting any vertex of A to v_1, v_2, v_3.

We want A to be a clique, so we want to prevent the case that a and b are not adjacent. But the non-adjacency of a and b yields a forbidden graph G_9, as shown

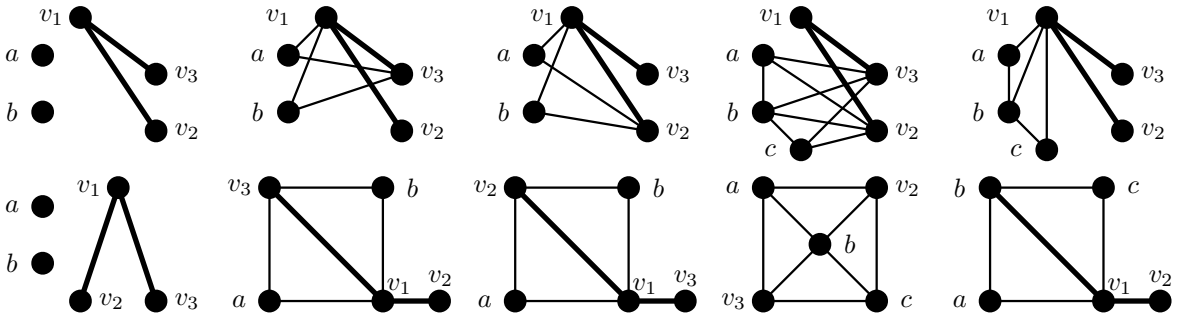


A missing edge $\{a, b\}$ in A creates a G_9

A missing edge $\{a, b\}$ in B creates a G_6

A missing edge $\{a, b\}$ in C creates a G_6

FIG. 6.9. Forbidden graphs created by missing edges in A, B or C.



A missing edge $\{a, b\}$ in E creates a G_5

A missing edge $\{a, b\}$ in F creates a G_7

A missing edge $\{a, b\}$ in G creates a G_7

A $K_{1,2}$ on $\{a, b, c\}$ in D creates a G_9

A $K_{1,2}$ on $\{a, b, c\}$ in H creates a G_7

FIG. 6.10. Forbidden graphs created by missing edges in E, F or G and by a $K_{1,2}$ in D or H.

below – the lower left graph in Figure 6.9 is the same graph as the upper right one; it is only redrawn to make its isomorphism to G_9 more obvious. This proves that A induces a clique in G'.

Figures 6.9 and 6.10 show the forbidden subgraphs created by a missing edge in A, B, C, E, F, G, with each forbidden situation (or just a part of it) redrawn in the way described above. By the assumptions, no forbidden subgraph is contained in G', so we conclude that all those groups induce cliques in G'.

The groups D and H need not be cliques; we just want each of them to be a disjoint union of cliques. We prove this by showing that if any of D, H contains a $K_{1,2}$ (and is not a disjoint union of cliques), it creates a forbidden subgraph. These situations are the last two cases of Figure 6.10. So D and H are disjoint unions of cliques.

We proceed by showing that the edges between vertices of different groups are as shown in Figure 6.11. We shall for simplicity speak about adjacency of whole cliques instead of single vertices.

Definition 6.9. We say that two cliques K_1 and K_2 are *adjacent* if all vertices of K_1 are adjacent to all vertices of K_2. We say that two cliques K_1 and K_2 are *non-adjacent* if no vertex of K_1 is adjacent to any vertex of K_2.

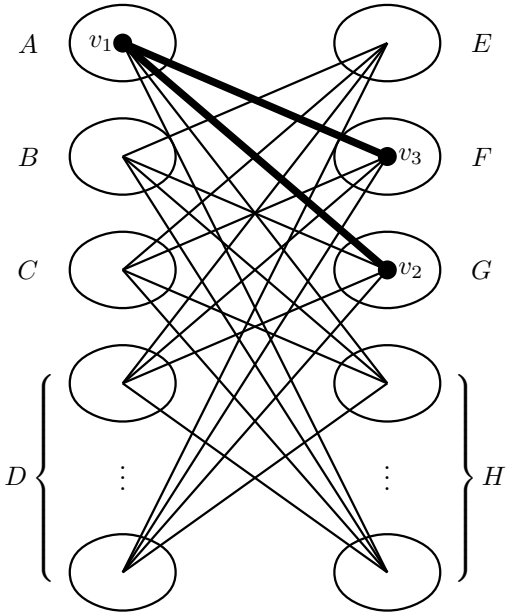


FIG. 6.11. Cliques and edges between them in a graph switching-equivalent to a $K_{1,2}$-free graph.

The desired situation (depicted in Figure 6.11) is as follows:

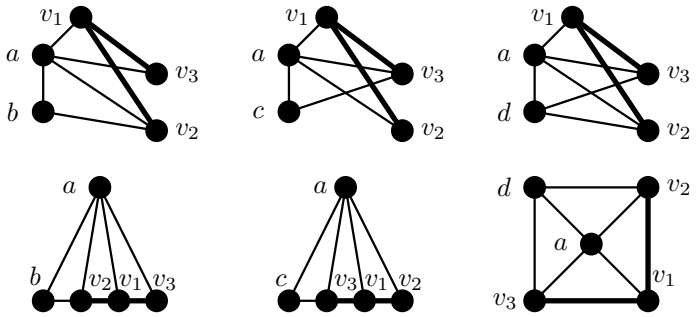
- Any group of A, B, C is adjacent to all groups on the right side except for E, F, G, respectively, to which it is non-adjacent.
- Any group of E, F, G is adjacent to all groups on the left side except for A, B, C, respectively, to which it is non-adjacent.
- The groups on the left side are pairwise non-adjacent. The groups on the right side are pairwise non-adjacent as well.
- Every clique in D is adjacent to all cliques in H except for at most one clique, to which it is non-adjacent.
- Every clique in H is adjacent to all cliques in D except for at most one clique, to which it is non-adjacent.

Moreover, the vertex v_1 has the same neighborhood (in the rest of the graph) as vertices of A, and is adjacent to all of them; so we can add v_1 to the clique A while keeping all the desired adjacency properties of A. Analogously, we add v_2 to F and v_3 to G.

By examining all the possible cases, we show that an unwanted, or wanted but missing edge would create a forbidden subgraph. Figures 6.12 and 6.13 show bad edges between A and the other groups, and the forbidden graphs caused. Figures 6.14 and 6.15 show bad edges between B and the other groups; the cases for C and the other groups are almost the same (only with v_2 and v_3 swapped). Figure 6.16 shows bad edges between D and E, F, or G.

The forbidden edge situations between D and H are slightly more complicated. We show that no two non-adjacent vertices in D (thus, belonging to two distinct cliques in D) can be both non-adjacent to the same vertex of H (and symmetrically). The converse would yield a G_6, as shown in Figure 6.17.

Moreover, no two adjacent vertices in D (thus, belonging both to the same clique) can have different neighborhood in H (and symmetrically). Otherwise we get a G_{10} or

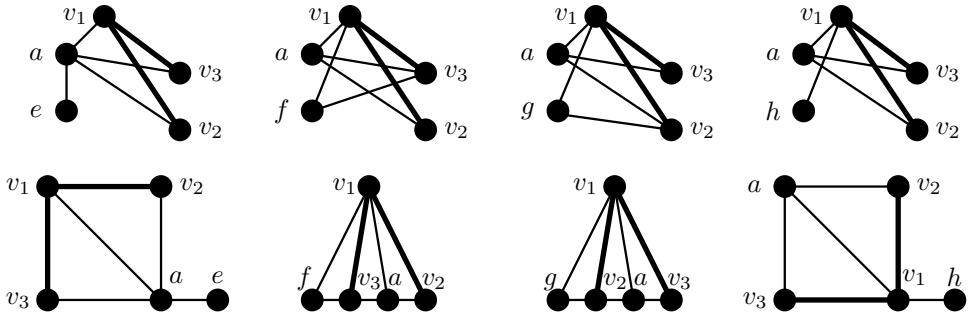


An edge $\{a, b\}$ with $a \in A$, $b \in B$ creates a G_3

An edge $\{a, c\}$ with $a \in A$, $c \in C$ creates a G_3

An edge $\{a, d\}$ with $a \in A$, $d \in D$ creates a G_9

FIG. 6.12. Bad edges between A and B, C, D.



An edge $\{a, e\}$ with $a \in A$, $e \in E$ creates a G_7

A non-edge $\{a, f\}$ with $a \in A$, $f \in F$ creates a G_3

A non-edge $\{a, g\}$ with $a \in A$, $g \in G$ creates a G_3

A non-edge $\{a, h\}$ with $a \in A$, $h \in H$ creates a G_7

FIG. 6.13. Bad edges between A and E, F, G, H.

a G_7, as shown in Figure 6.17. So all vertices of one clique of D or H have the same neighborhood in the rest. And for every clique in D or H, there is at most one clique in H or D non-adjacent to it, respectively. This proves that the adjacencies between D and H are as depicted in Figure 6.11.

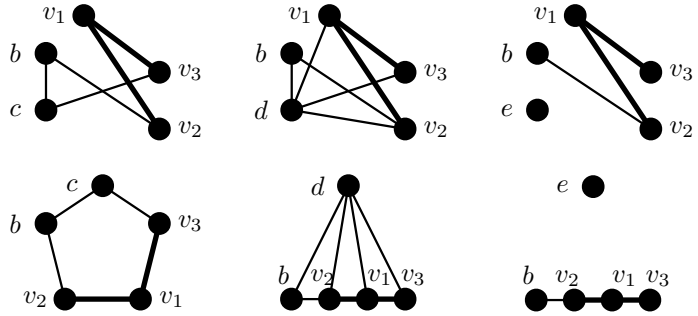
Figures 6.18 and 6.19 represent the remaining bad cases for edges between E, F, G, and H. That finishes the case analysis of edges in G'.

It remains to prove that the graph

$$G^* = S(G', A \cup B \cup C \cup D)$$

is a disjoint union of cliques. That is true by the following argument. The groups A and E are non-adjacent in G', so in G^* they are adjacent, and form a clique together. A is non-adjacent to the rest of G^*, and so is E. The same holds for B and F, C and G. Since every clique of D is non-adjacent to at most one clique of H, in G^* we have the contrary: every clique of D is adjacent to at most one clique of H, together with which it forms a clique in G^* (non-adjacent to the rest of G^*). So $G^* = S(G', A \cup B \cup C \cup D)$ is really a disjoint union of cliques, which is a $K_{1,2}$-free graph. □

Proof. (of Theorem 6.5) The theorem follows immediately from Lemmas 6.6, 6.7, and 6.8. □

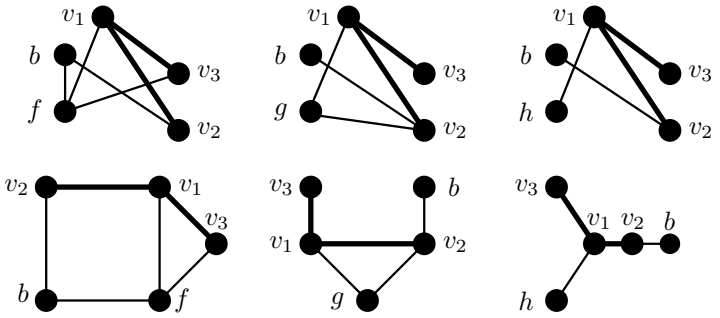


An edge $\{b, c\}$ $b \in B$, $c \in C$ creates a G_1

An edge $\{b, d\}$ $b \in B$, $d \in D$ creates a G_3

A non-edge $\{b, e\}$ $b \in B$, $e \in E$ creates a G_4

FIG. 6.14. Bad edges between B and C, D, E. The cases for C are analogous.

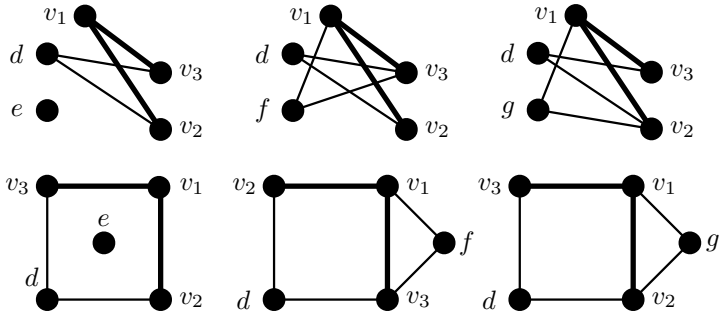


An edge $\{b, f\}$ $b \in B$, $f \in F$ creates a G_{10}

A non-edge $\{b, g\}$ $b \in B$, $g \in G$ creates a G_2

A non-edge $\{b, h\}$ $b \in B$, $h \in H$ creates a G_6

FIG. 6.15. Bad edges between B and F, G, H. The cases for C are analogous.



A non-edge $\{d, e\}$ $d \in D$, $e \in E$ creates a G_8

A non-edge $\{d, f\}$ $d \in D$, $f \in F$ creates a G_{10}

A non-edge $\{d, g\}$ $d \in D$, $g \in G$ creates a G_{10}

FIG. 6.16. Bad edges between D and E, F, and G.

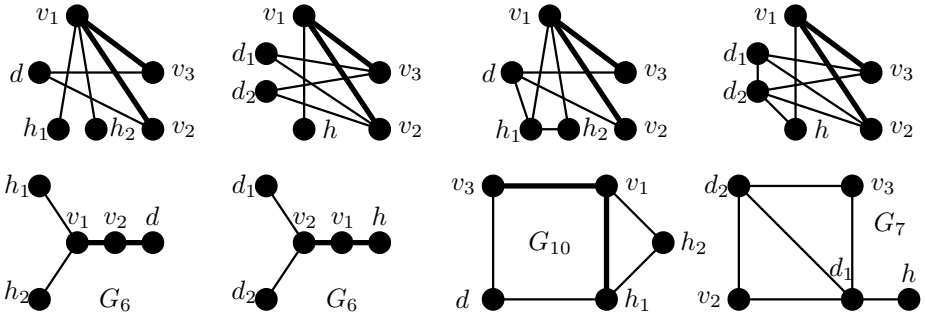
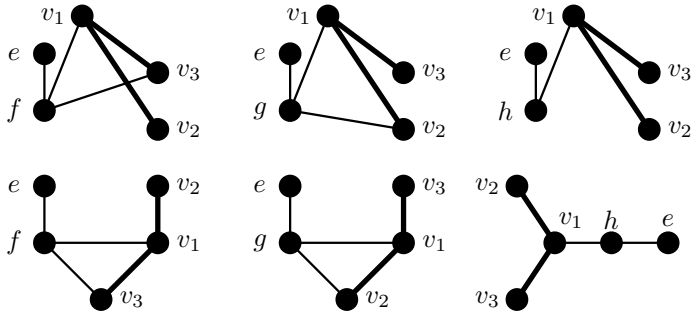


FIG. 6.17. Bad edges between D and H.

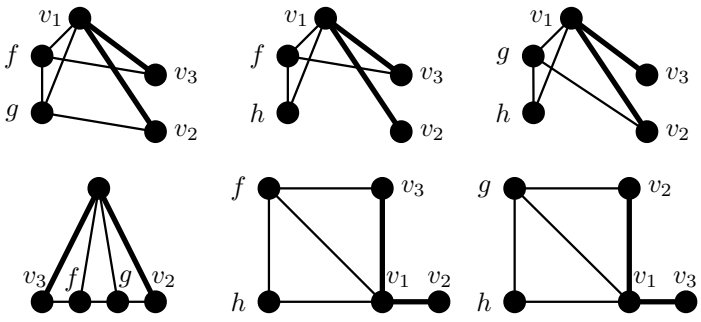


An edge $\{e, f\}$
$e \in E$, $f \in F$
creates a G_2

An edge $\{e, g\}$
$e \in E$, $g \in G$
creates a G_2

An edge $\{e, h\}$
$e \in E$, $h \in H$
creates a G_6

FIG. 6.18. Bad edges between E and F, G, H.



An edge $\{f, g\}$
$f \in F$, $g \in G$
creates a G_3

An edge $\{f, h\}$
$f \in F$, $h \in H$
creates a G_7

An edge $\{g, h\}$
$g \in G$, $h \in H$
creates a G_7

FIG. 6.19. Bad edges between F, G, and H.

7. Concluding remarks

In Chapter 4, we have introduced several known NP-complete problems related to switching, mostly the problems $S(P)$ that are NP-complete. Then we have extended them by proving that S("containing a clique of size at least cn") is NP-complete even for a fixed c. In Section 4.6, we have shown the hardness of approximation of the switching versions of maximum clique and chromatic number.

In Chapter 5, we have examined the complexity of S("being H-free") for several fixed graphs H. Polynomial-time decision algorithms are known for this problem, if H has at most three vertices or is isomorphic to a P_4. In Section 5.2 we have shown that if H is isomorphic to a claw, then the problem is polynomial as well. Thus all known results for particular graphs H give polynomial-time algorithms, which yields the following problem.

Problem 7.1. *Is there a graph H such that the problem S("being H-free") is NP-complete?*

The proof of Theorem 5.3 in Section 5.2 uses a reduction to a linear equation system over $GF(2)$. A similar reduction works for H isomorphic to a $K_{1,2}$; however, for $K_{1,2}$ there is already a more efficient algorithm due to Kratochvíl at al. [15] (see Section 3.4). It might be interesting to find out if a similar approach – possibly using a larger finite field instead of $GF(2)$ – works for some other graphs as well; we have not found any.

In Chapter 6, we deal with characterizations of various switching problems by forbidden induced subgraphs. We focus on those of S("being H-free") for several graphs H. Clearly, a characterization by a finite list of forbidden induced subgraphs yields a polynomial-time recognition algorithm. However, a polynomial-time recognition algorithm may exist even if the minimal list is infinite; but no such graph H is known.

The lists of minimal forbidden induced subgraphs are known for K_2 and I_2 (see Section 3.6), $K_{1,2}$ and K_1+K_2 (see Section 6.5), and P_4 (see Section 6.2). In Section 6.5, we have given the list for H isomorphic to $K_{1,2}$; it consists of ten graphs, each having five vertices.

In case of the triangle K_3 and the claw $K_{1,3}$, the corresponding polynomial-time recognition algorithms exist (see Section 3.5 and Section 5.2), but the lists of forbidden induced subgraphs are not known. Moreover, it seems that there is not a reasonably short list for either of them. According to the computational results, both lists soon grow to large size, which we find interesting in view of the fact that the list for $K_{1,2}$ has ten graphs only, and the graphs K_3 and $K_{1,2}$ differ just in one edge. Also, the minimal forbidden induced subgraphs for $K_{1,2}$ all have five vertices; but for K_3 we have found hundreds of minimal forbidden induced subgraphs on nine or more vertices, and even one on fifteen vertices (shown in Figure 7.20). For $K_{1,3}$ there are also hundreds of minimal forbidden induced subgraphs on nine vertices; the maximum number of vertices for which we have found one is twelve.

This indicates that if an upper bound M for the number of vertices of a minimal forbidden induced subgraph was found, a naive search through all graphs on at most M

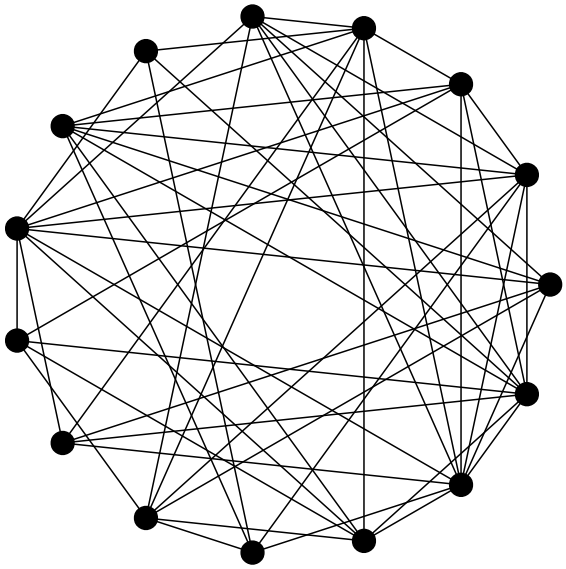


FIG. 7.20. One of hundreds of minimal forbidden induced subgraphs for $H = K_3$.

vertices would be nearly impossible. However, the lists may still be finite; the following problem remains open.

Problem 7.2. *Is there a graph H such that the list of minimal forbidden induced subgraphs for the property "being switching-equivalent to an H-free graph" is infinite?*

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof verification and the hardness of approximation problems*, Journal of ACM **45(3)** (1998), pp. 501–555. An extended abstract appeared in FOCS 1992.

[2] C. J. Colbourn, D. G. Corneil, *On deciding switching equivalence of graphs*, Discrete Appl. Math **2** (1980), pp. 181–184.

[3] A. Ehrenfeucht, J. Hage, T. Harju, G. Rozenberg, *Pancyclicity in switching classes*, Inform. Process. Letters **73** (2000), pp. 153–156.

[4] A. Ehrenfeucht, J. Hage, T. Harju, G. Rozenberg, *Complexity issues in switching classes*, in: Theory and Application to Graph Transformations, LNCS 1764, Springer-Verlag (2000), pp. 59–70.

[5] U. Feige, J. Kilian, *Zero Knowledge and the Chromatic Number*, in: Proc. of the 11'th IEEE Conference on Computational Theory (1996), pp. 278–287.

[6] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[7] J. Hage, *Structural Aspects Of Switching Classes* (PhD thesis), Leiden Institute of Advanced Computer Science, 2001. Available online at `http://www.cs.uu.nl/people/jur/2s.html`.

[8] J. Hage, T. Harju, E. Welzl, *Euler graphs, triangle-free graphs and bipartite graphs in switching classes*, in: Proceedings ICGT 2002, LNCS 2505, Springer-Verlag (2002), pp. 148–160.

[9] J. Hage, T. Harju, *A characterization of acyclic switching classes of graphs using forbidden subgraphs*, SIAM J. Discrete Math. **18** (2004), pp. 159–176.

[10] R. B. Hayward, *Recognizing P_3-structure: A switching approach*, J. Combin. Th. Ser. B **66** (1996), pp. 247–262.

[11] R. B. Hayward, S. Hougardy, B. A. Reed, *Polynomial time recognition of P_4-structure*, Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (2002), pp. 382–389.

[12] J. Håstad, *Clique is hard to approximate within $n^{1-\epsilon}$*, Acta Mathematica **85** (1999), pp. 105–142. An extended abstract appeared in FOCS 1996.

[13] A. Hertz, *On perfect switching classes*, Discrete Appl. Math. **94** (1999), pp. 3–7.

[14] J. Kratochvíl, *Perfect codes and two-graphs*, Comment. Math. Univ. Carolin. **30** (1989), pp. 755–760.

[15] J. Kratochvíl, J. Nešetřil, O. Zýka, *On the computational complexity of Seidel's switching*, Proc. 4th Czech. Symp., Prachatice 1990, Ann. Discrete Math. **51** (1992), pp. 161–166.

[16] J. Kratochvíl, *Complexity of hypergraph coloring and Seidel's switching*, Graph Theoretic Concepts in Computer Science (H. Bodlaender, ed.), Proceedings WG 2003, Elspeet, June 2003, LNCS 2880, Springer Verlag, 2003.

[17] C. L. Mallows, N. J. A. Sloane, *Two-graphs, switching classes and Euler graphs are equal in number*, SIAM J. Appl. Math **28** (1975), pp. 876–880.

[18] R. W. Robinson, *Enumeration of Euler graphs*, Proof Techniques in Graph Theory, F. Harary, ed., Academic Press, N. Y., 1969.

[19] J. J. Seidel, *Graphs and two-graphs*, in: Proc. 5th. Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Winnipeg, Canada (1974).

[20] J. J. Seidel, *A survey of two-graphs*, Teorie combinatorie, Atti Conv. Lincei, Vol 17, Academia Nazionale dei Lincei, Rome (1973), pp. 481–511.

[21] J. J. Seidel, D. E. Taylor, *Two-graphs, a second survey*, Algebraic methods in graph theory, Vol. II, Conf. Szeged 1978, Colloq. Math. Janos Bolyai **25** (1981), pp. 689–711.

[22] J. J. Seidel, *More about two-graphs*, Combinatorics, graphs and complexity, Proc. 4th Czech. Symp., Prachatice 1990, Ann. Discrete Math. **51** (1992), pp. 297–308.

[23] V. V. Vazirani, *Approximation Algorithms*, Springer-Verlag, 2001.

[24] T. Zaslavsky, *Signed graphs*, Discrete Applied Math. **4** (1982), pp. 47–74. Erratum on p. 248 of volume 5.