Discrete Mathematics and Theoretical Computer Science DMTCS vol. **13:2**, 2011, 19–42

Parameterized Problems Related to Seidel's Switching

Eva Jelínková[1†] Ondřej Suchý[3‡§] Petr Hliněný[4§] Jan Kratochvíl[1,2§]

[1]*Department of Applied Mathematics*[¶] [2] *Institute for Theoretical Computer Science*
Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic
[3]*Cluster of Excellence "Multimodal Computing and Interaction", Saarland University, Saarbrücken, Germany*
[4]*Faculty of Informatics, Masaryk University, Brno, Czech Republic*

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 paper, we continue the study of computational complexity aspects of Seidel's switching, concentrating on Fixed Parameter Complexity. Among other results we show that switching to a graph with at most k edges, to a graph of maximum degree at most k, to a k-regular graph, or to a graph with minimum degree at least k are fixed parameter tractable problems, where k is the parameter. On the other hand, switching to a graph that contains a given fixed graph as an induced subgraph is $W[1]$-complete. We also show the NP-completeness of switching to a graph with a clique of linear size, and of switching to a graph with small number of edges. A consequence of the latter result is the NP-completeness of Maximum Likelihood Decoding of graph theoretic codes based on complete graphs.

Keywords: Seidel's Switching, Computational Complexity, Parameterized Complexity

1 Introduction

The concept of Seidel's switching (for a formal definition, see Subsection 2.2) was introduced by a Dutch mathematician J. J. Seidel in connection with regular structures, such as systems of equiangular lines, strongly regular graphs, or the so-called two-graphs. For more structural properties of two-graphs, cf. [Sei74, Sei76, ST81]. Since then, switching has been studied by many others. Apart from the algebraic structures, consequences of switching arise in other research fields as well; for example, Seidel's switching plays an important role in Hayward's polynomial-time algorithm for solving the P_3-structure recognition problem [Hay96].

[‖]eva@kam.mff.cuni.cz, suchy@mmci.uni-saarland.de, hlineny@fi.muni.cz, honza@kam.mff.cuni.cz

[†]Part of work done while visiting the Icelandic Centre of Excellence in Theoretical Computer Science, Reykjavík, Iceland.
[‡]Most of the work done while on the Dep. of Applied Math. and Inst. for Theoretical Computer Science, Charles University.
[§]Supported by grant 1M0545 of the Czech Ministry of Education.
[¶]Supported by project 1M0021620838 of the Czech Ministry of Education.

As proved by Colbourn and Corneil [CC80] (and independently by Kratochvíl et al. [KNZ92]), deciding whether two given graphs are switching-equivalent is an isomorphism-complete problem.

Several authors have considered the complexity of deciding if a given graph can be switched to a graph having some particular property. It was noted already in [KNZ92] that the complexity of such problems is in no correlation with the complexity of the property itself: Kratochvíl et al. [KNZ92] proved that any graph is switching-equivalent to a graph containing a Hamiltonian path, and it is polynomial to decide if a graph is switching-equivalent to a graph containing a Hamiltonian cycle (as proved also by Ehrenfeucht et al. [EHHR00b]). However, the problems to decide if a graph itself contains a Hamiltonian path or cycle are well known to be NP-complete [GJ79]. On the other hand, Kratochvíl [Kra03] showed NP-completeness of deciding if a given graph can be switched to a regular one, while deciding if a given graph is regular is obviously polynomial-time solvable. Somewhat surprisingly, the above mentioned switching to a regular graph, and switching to a graph with large clique number remain until today the only nontrivial hard cases.

Polynomial-time algorithms are known for switching to a triangle-free graph [Hay96, HHW02], to a claw-free graph [JK08], to an eulerian graph [HHW02], to a bipartite graph [HHW02], and to a planar graph [EHHR00a, Kra03].

In this paper, we initiate the study of the complexity of Seidel's switching from the Fixed Parameter Complexity point of view. The polynomial algorithm for testing switchability to a planar graph is based on the existence of a vertex of degree at most 5 in any planar graph. This naturally suggests to consider the problem of switching to a k-degenerate graph. For fixed k, the problem is solvable in time $\mathcal{O}(n^{k+3})$ [EHHR00a], but the fixed-parameter complexity, when parameterized by k, is still open. We determine the fixed-parameter complexity of several closely related variants: we prove that switching to a graph of maximum degree at most k is fixed parameter tractable, as well as of switching to a graph of minimum degree at least k, and of switching to a k-regular graph. All these results are presented in Section 3.

It is easy to see that every graph with n vertices can be switched to a graph with maximum degree at most $n/2$. Indeed, consider a switching-equivalent graph with the minimum number of edges; switching a vertex of degree greater than $n/2$ would reduce the number of edges. This argument also gives an algorithm for constructing a switch with all degrees at most $n/2$. However, this does not provide a polynomial-time algorithm for constructing the switch with minimum number of edges. And indeed, answering a long standing open problem [Hag01], we prove in Section 4 that deciding if a given graph can be switched to a graph with at most k edges is NP-complete. We also show a connection to coding theory in Subsection 4.2. Of course, a graph with at most k edges has all vertex degrees also at most k, and hence it is not surprising that from the Fixed Parameter Complexity point of view this question is easy. We will describe an FPT algorithm with a considerably better running time in Subsection 4.3.

Switching to a graph containing a large clique is another computationally difficult problem. It was shown NP-complete in [EHHR00a]. We show in Section 5 that switching to a graph with $\omega(G) \geq cn$ is NP-complete for every constant $0 < c < 1$. The more involved case of irrational c is proved by applying the PCP theorem about inapproximability of the 3SAT problem. In Section 6 we show that this question is W[1]-hard when considered parameterized by the size of the clique.

Conference versions of some of the results contained in the present paper have appeared in conference proceedings [Jel08] and [Suc08].

2 Basic Definitions

2.1 Preliminaries

In this paper, we use the standard graph theoretic notation. Unless defined otherwise, by n we denote the number of vertices of the currently discussed graph. If $G = (V, E)$ and $A \subseteq V$ then $G[A]$ denotes the graph $(A, E \cap \binom{A}{2})$ induced by the vertices in A, and $\overline{G}$ denotes the complement $(V, \binom{V}{2} \setminus E)$ of the graph G. The 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 path with n vertices is denoted by P_n, and a graph with n vertices and no edges is called *discrete* and denoted by I_n. The symmetric difference of sets A and B is denoted by $A \triangle B$.

A given parameterized problem is *fixed-parameter tractable (FPT)* with respect to a parameter k, if there is an algorithm solving the problem in $f(k) \cdot n^c$ time for some computable function f and a constant c. Similarly, if there is an algorithm running in time $n^{g(k)}$ for some computable function g, then we say that the problem is in the class XP (both these notions were introduced by Downey and Fellows [DF99]).

To catch the fundamental difference between $f(k) \cdot n^c$ and $n^{g(k)}$ running time, parameterized hardness theory was introduced. A parameterized problem Q is *FPT-reducible* to a parameterized problem Q' if there exists an algorithm of running time $f(k)|x|^{O(1)}$ that on an instance (x, k) of Q produces an instance $(x', g(k))$ of Q' such that (x, k) is a YES-instance of Q if and only if $(x', g(k))$ is a YES-instance of Q', where the functions f and g depend only on k. Downey and Fellows [DF99] introduced the class W[1] as the basic class of fixed-parameter intractable problems — W[1]-hard problems are believed not to have algorithms running in time $f(k) \cdot n^c$ (otherwise, the "exponential time hypothesis" would fail). A parameterized problem Q is W[1]-hard if every problem in W[1] is FPT-reducible to Q, and W[1]-complete, if it is W[1]-hard and in W[1]. See [DF99, FG06, Nie06] for the definition of the class W[1] and more details on parameterized complexity.

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 the 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.,*

$$\begin{aligned} V_{S(G,v)} &= V_G \\ E_{S(G,v)} &= \big(E_G \setminus \{xv : x \in V_G\}\big) \cup \{xv : x \in V_G,\ x \neq v,\ xv \notin E_G\}. \end{aligned}$$

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 generalization of switching to vertex subsets of G.

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

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

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.*

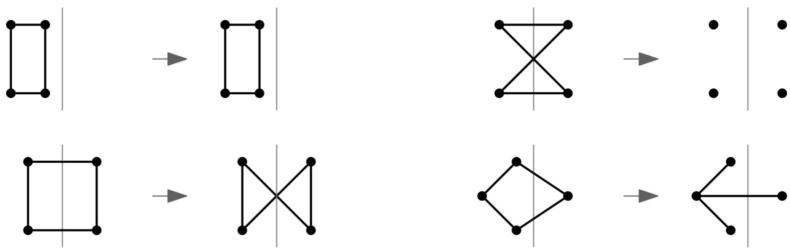

Fig. 1: Seidel's switches of C_4. Switched vertices are drawn left of the vertical line.

To give an example; Fig. 1 depicts all the possible switches of C_4 (up to isomorphism). The vertices of the subset A are drawn left of the vertical line. It can be seen that all graphs switching-equivalent to C_4 are C_4 itself, the *claw* $K_{1,3}$ and the discrete graph I_4.

From the case of C_4 in Fig. 1, the reader may already deduce some basic observations about Seidel's switching. Various properties, as shown in the following lemma, can be found in [Hag01].

Lemma 2.3 *Let G be a graph and $A, B \subseteq V$ two subsets of its vertices. Then*

- $S(G, A)[A] = G[A]$,
- $S(G, A) = S(G, V \setminus A)$,
- $S(G, \emptyset) = S(G, V) = G$,
- $S(S(G, A), B) = S(S(G, B), A) = S(G, A \triangle B)$,
- $S(S(G, A), A) = G$,
- $S(\overline{G}, A) = \overline{S(G, A)}$.

Lemma 2.4 (Ehrenfeucht et al. [EHHR00a]) *Let $G = (V, E)$ be a graph, $u \in V$ and $X \subseteq V$. If $u \notin X$, then there exists a unique graph $H \in [G]$ such that the neighbors of u in H are the vertices in X. Namely, it is the graph $S(G, N_G(v) \triangle X)$.*

3 Optimizing Vertex Degrees

We examine the following problems:

SWITCH-k-SMALL-DEGS **Input:** A graph $G = (V, E)$, an integer k **Question:** Is there a subset of vertices $A \subseteq V$ such that in the graph $S(G, A)$, the degree of each vertex is at most k?

SWITCH-k-LARGE-DEGS **Input:** A graph $G = (V, E)$, an integer k **Question:** Is there a subset of vertices $A \subseteq V$ such that in the graph $S(G, A)$, the degree of each vertex is at least k?

SWITCH-k-REGULAR **Input:** A graph $G = (V, E)$, an integer k **Question:** Is there a subset of vertices $A \subseteq V$ such that the graph $S(G, A)$ is k-regular?

Note that there are pairs (G, k) that form a YES-instance for both SWITCH-k-SMALL-DEGS and SWITCH-k-LARGE-DEGS but a NO-instance for SWITCH-k-REGULAR, e.g. $(C_4, 1)$, see Fig. 1.

It is easy to see that, for every fixed k, graphs with all vertices of degrees at most k form a class to which the following general result applies, and k-regular graphs form such a class as well. The Theorem was proved by Ehrenfeucht et al. [EHHR00a] and also independently (in a slightly weaker form) by Kratochvíl [Kra03].

Theorem 3.1 *Let $\mathcal{P}$ be a graph property that can be decided in time $\mathcal{O}(n^a)$ for an integer a. Let every graph with $\mathcal{P}$ contain a vertex of degree at most $d(n)$. Then the problem if an input graph is switching-equivalent to a graph with $\mathcal{P}$ can be decided in time $\mathcal{O}(n^{d(n)+1+\max(a,2)})$.*

In particular, Theorem 3.1 implies the existence of an $O(n^{k+3})$-time algorithm for deciding if a given graph can be switched to a graph with all vertex degrees at most k, or to a k-regular graph.

That means that the problem SWITCH-k-REGULAR, which is known to be NP-complete [Kra03], is in the class XP when considered parameterized by k. Hence it is natural to examine the parameterized complexity of the problems with respect to this obvious parameter. This is done in Subsection 3.1 for large degrees, in Subsection 3.2 for small degrees, and in Subsection 3.3 for regular graphs.

3.1 Parameterized Complexity of SWITCH-k-LARGE-DEGS

The case of SWITCH-k-LARGE-DEGS is simple due to the following folklore lemma.

Lemma 3.2 *Let $G = (V, E)$ be a graph on n vertices. Then there is a set $A \subseteq V_G$ such that in the graph $S(G, A)$, degrees of all vertices are at least $\lfloor n/2 \rfloor$.*

Proof: Suppose on contrary that there is no such set. Moreover assume that A_0 is such that $S(G, A_0)$ has the highest number of edges among all $S(G, A)$. Suppose v is a vertex of degree $d < \lfloor n/2 \rfloor$ in $S(G, A_0)$. Then switching v in $S(G, A_0)$ destroys d edges while introducing $n - d - 1 \geq \lfloor n/2 \rfloor$ edges. Hence $S(G, A_0 \triangle \{v\})$ has more edges than $S(G, A_0)$, a contradiction.

Note that we can also actually find the appropriate set in cubic time by iteratively switching the vertices of small degree, since each such iteration increases the number of edges and can be done in linear time with a suitable graph representation. □

Corollary 3.3 SWITCH-k-LARGE-DEGS *is fixed-parameter tractable—it can be decided in time $O(4^k \cdot k^2 + n)$.*

Proof: If $k \leq n/2$, then by Lemma 3.2, any graph on n vertices can be switched to a graph with all vertices of degree at least k. Hence, the answer is trivially yes.

Otherwise $n < 2k$, and we may solve the instance by trying all possible 2^{n-1} switches of the input graph, and this takes time at most $n^2 \cdot 2^{n-1} \leq k^2 \cdot 2^{2k}$. We allow linear time for the reading of the whole input. □

3.2 Parameterized Complexity of SWITCH-k-SMALL-DEGS

In this section, we present the best known algorithm for the parameterized version of SWITCH-k-SMALL-DEGS. The ideas of the algorithm are the crucial ones in the proofs of the tractability of problems in further sections of the paper.

Before we present the algorithm, we observe that we can assume that the input graph contains an isolated vertex v_0. If it does not, we can switch the neighborhood of a vertex of minimum degree to obtain an isolated vertex according to Lemma 2.4. As one vertex can be switched in time $O(n)$, we can obtain the isolated vertex in time $O(d \cdot n)$ which is upperbounded by $O(m)$, where d is the minimum degree and m is the number of the edges of the original graph. The answer remains unchanged due to Lemma 2.3.

Let G be the input graph. We seek a set $A \subseteq V_G$ such that $S(G, A)$ has all vertex degrees at most k. Since A is a solution if and only if $V \setminus A$ is, we may assume without loss of generality that $v_0 \notin A$. These two things together give us that $N_{S(G,A)}(v_0) = A$. Thus necessarily $|A| \leq k$.

This means that in total we can switch at most k vertices. During the run of the algorithm, we switch certain vertices one by one, and by l we denote the number of the vertices we can still switch. At the beginning we set $l := k$.

Now we introduce two simple rules that solve the "big" instances:

Lemma 3.4 (Rule 1) *We must switch every vertex v of degree greater than $k + l$ in order to obtain a solution.*

Proof: By contradiction. Suppose that A determines a switch solving the problem and that $v \notin A$. Vertex v has at least $k + l + 1$ neighbors in G. By switching v could lose at most $|A|$ of them and thus at most l. So it still has at least $k + 1$ neighbors. Switching A does not solve the problem, a contradiction. □

Lemma 3.5 (Rule 2) *No vertex v of degree less than $n - k - l$ can be switched to obtain a solution.*

This lemma is proved similarly as the previous one, so we omit the proof.

The following statement is an easy corollary of the previous two lemmas:

Corollary 3.6 (Boundary) *If $n \geq 2k + 2l + 1$, then at least one of the rules 1 and 2 applies on each vertex.*

That means we have already obtained a kernel with $4k$ vertices (i.e. the instances with more than $4k$ vertices can be solved efficiently, while only those with at most $4k$ vertices are really hard) and thus the problem is in FPT.

As we want to give a better bound on the running time of the algorithm, we concentrate on the instances of sizes $n = 2k + 2p + 2$ or $n = 2k + 2p + 1$ for some $0 \leq p \leq k - 1$. In this case, after switching $k - p$ vertices (i.e., if at most p more vertices can be switched) at least one of the rules applies to each vertex and, hence, it is possible to decide which vertices should be switched and which should not. The algorithm is as follows:

1. Set $l := k$
2. For every vertex v (except v_0)
   - Check which of the rules 1 and 2 applies on v
   - If both rules apply answer NO and quit.

- If rule 1 applies switch v and decrease l
- If $l < 0$ answer NO and quit.

3. If $n > 4k$ then check if all the vertices have degree at most k in this switch and answer.
4. If $n \leq 4k$ then try all the possibilities A to choose up to $l - p := l - \lfloor \frac{1}{2}(n - 2k - 1) \rfloor$ vertices that can be switched and that were not switched already. For each choice of A switch according to A, apply the rules once again and check whether we obtained the desired solution.

Remark 3.7 *The algorithm can be also used to enumerate all the graphs with the desired property.*

The correctness of the algorithm follows immediately from Lemmas 3.4, 3.5 and Corollary 3.6.

To count the running time of the algorithm we must bound the number $\sum_{i=0}^{k-p} \binom{n-1}{i}$ of the candidate sets in step 4. It is easy to observe that this is always at most $\binom{2k+2p+1}{k-p} \cdot k$, so we have to search for the biggest number among

$$\binom{2k+1}{k}, \binom{2k+3}{k-1}, \dots, \binom{2k+2p+1}{k-p}, \dots, \binom{4k-3}{2}, \binom{4k-1}{1}.$$

We compare two neighboring terms $z_k(p) := \binom{2k+2p+1}{k-p}$ and $z_k(p+1) = \binom{2k+2p+3}{k-p-1}$. Since

$$z_k(p) \quad = \quad \binom{2k+2p+1}{k-p} = \frac{(2k+2p+1)\dots(k+3p+5)(k+3p+4)(k+3p+3)(k+3p+2)}{(k-p)(k-p-1)\dots 2 \cdot 1},$$

while

$$z_k(p+1) \quad = \quad \binom{2k+2p+3}{k-p-1} = \frac{(2k+2p+3)(2k+2p+2)(2k+2p+1)\dots(k+3p+5)}{(k-p-1)\dots 2 \cdot 1}$$

we know that the term $z_k(p)$ is less than (equal, greater than) the term $z_k(p+1)$ if and only if the left side of a cubic equation $(k+3p+4) \cdot (k+3p+3) \cdot (k+3p+2) = (2k+2p+3) \cdot (2k+2p+2) \cdot (k-p)$ is less than (equal, greater than) the right side, respectively. The equation (considered with parameter k) has only one real root

$$\begin{aligned} p_0 = {} & \frac{1}{93} \cdot (-31k - 91 + (30752k^3 + 47616k^2 + 19518k + 1403 + \\ + {} & 186\sqrt{3}\sqrt{7936k^6 + 23808k^5 + 24352k^4 + 7776k^3 - 3541k^2 - 3424k - 975})^{1/3} \\ + {} & (496k^2 + 620k + 469) \cdot (30752k^3 + 47616k^2 + 19518k + 1403 + \\ + {} & 186\sqrt{3}\sqrt{7936k^6 + 23808k^5 + 24352k^4 + 7776k^3 - 3541k^2 - 3424k - 975})^{-1/3}), \end{aligned}$$

which is $p_0 = 0.22298k + o(k)$. Since the term p^3 appears positively on the left side, while negatively on the right side of the equation, the left side is less than the right side if and only if $p < p_0$. Hence $z_k(\lceil p_0 \rceil - 1)$ is less than $z_k(\lceil p_0 \rceil)$ while $z_k(\lceil p_0 \rceil)$ is already greater than $z_k(\lceil p_0 \rceil + 1)$. Thus for large k the biggest term is $z_k(\lceil p_0 \rceil) = \binom{2\lceil 1.2230k \rceil + 1}{\lfloor 0.7770k \rfloor}$. Using the standard approximation

$$e\left(\frac{n}{e}\right)^n \leq n! \leq ne\left(\frac{n}{e}\right)^n$$

for the factorials involved, we have:

$$\begin{aligned}
&\binom{2\lceil 1.2230k\rceil+1}{\lfloor 0.7770k\rfloor} \leq \binom{2.4461k}{0.7770k} \leq \frac{e(2.4461k)\left(\frac{2.4461k}{e}\right)^{2.4461k}}{e\left(\frac{0.7770k}{e}\right)^{0.7770k}\cdot e\left(\frac{1.6691k}{e}\right)^{1.6691k}} = \\
= &\ \frac{2.4461k}{e}\frac{(2.4461)^{2.4461k}}{(0.7770)^{0.7770k}\cdot(1.6691)^{1.6691k}} = \\
= &\ \frac{2.4461k}{e} exp(k(2.4461\ln 2.4461 - 0.7770\ln 0.7770 - 1.6691\ln 1.6691)) \leq \\
\leq &\ 4.6135^k,
\end{aligned}$$

where the inequalities hold for k big enough.

We summarize our results in the following theorem:

Theorem 3.8 *The problem* SWITCH-k-SMALL-DEGS *is linear fixed-parameter tractable — it can be solved in time $O(4.614^k \cdot n + m)$.*

Proof: A graph G with an isolated vertex can be obtained in time $\mathcal{O}(m)$ by switching the neighborhood of the vertex of the smallest degree. Then we just test some candidate sets, each of the size at most k. It takes $\mathcal{O}(k \cdot n)$ time to switch the graph according to the candidate set and $\mathcal{O}(n)$ time to check if it is a solution. If $n > 4k$ then we have at most one candidate. If $n \leq 4k$ then we can have up to $\binom{2\lceil 1.2230k\rceil+1}{\lfloor 0.7770k\rfloor}\cdot k$ candidates, as we have counted before. This grows approximately as $4.6135^k \cdot k$. □

3.3 Parameterized Complexity of SWITCH-k-REGULAR

In an arbitrary k-regular graph, each vertex has degree at most k, which means that we can use our previous algorithm from Subsection 3.2 to decide the problem. We just enumerate all the degree $\leq k$ graphs and check if some of them is k-regular. But we will slightly improve the algorithm.

We always have to switch exactly k vertices in this case. Consider some ordering on $V \setminus \{v_0\}$. A k-vertex set can contain at most p vertices among the last p in this ordering. Hence it must contain at least $k-p$ vertices among the first $n-p-1$. In the case of $n \leq 4k$, the k-vertex sets determining the solution are searched by trying all size $k-p$ vertex subsets and then applying the rules to find the rest of the set. Due to the previous statement, we can find each solution searching the $k-p$ subsets only among the first $n-p-1$ vertices in the ordering.

Using this idea we can bound the number of candidate tests by $\binom{n-p-1}{k-p} \leq \binom{2k+p+1}{k-p}$. To find the bound for this, we again compare these terms for $0 \leq p \leq k-1$. The corresponding quadratic equation is $(k+2p+3)\cdot(k+2p+2) = (2k+p+2)\cdot(k-p)$, where one root is

$$p_1 = \frac{-5k-12+\sqrt{45k^2+60k+24}}{10},$$

and the second does not satisfy the condition $0 \leq p \leq k-1$. Asymptotically $p_1 = 0.170821k + o(k)$, which means that $\binom{\lceil 2.1709k\rceil+1}{\lfloor 0.8291k\rfloor}$ is the biggest term. Again using the standard factorial approximation we have:

$$\binom{\lceil 2.1709k\rceil+1}{\lfloor 0.8291k\rfloor} \leq \binom{2.1710k}{0.8291k} = \frac{2.1710k}{e}\frac{(2.1710)^{2.1710k}}{(0.8291)^{0.8291k}\cdot(1.3419)^{1.3419k}} =$$

$$= \frac{2.1710k}{e} exp(k(2.1710 \ln 2.1710 - 0.8291 \ln 0.8291 - 1.3419 \ln 1.3419)) \leq$$
$$\leq 4.2363^k,$$

for k large enough.

Our results are summarized by the following theorem:

Theorem 3.9 *The problem* SWITCH-k-REGULAR *is linear fixed-parameter tractable and can be solved in time* $\mathcal{O}\left(4.237^k \cdot n + m\right)$.

4 Optimizing the Number of Edges

The number of edges of a graph is another natural quantity to be optimized. We examine the following problems.

SWITCH-FEW-EDGES
Input: A graph $G = (V, E)$, an integer k
Question: Is there a subset of vertices $A \subseteq V$ such that in the graph $S(G, A)$, there are at most k edges?

SWITCH-MANY-EDGES
Input: A graph $G = (V, E)$, an integer k
Question: Is there a subset of vertices $A \subseteq V$ such that in the graph $S(G, A)$, there are at least k edges?

If k is a constant, there obviously is an $\mathcal{O}(n^{k+3})$-time algorithm for SWITCH-FEW-EDGES by Theorem 3.1. The problem SWITCH-MANY-EDGES becomes trivial in view of Lemma 3.2. For a parameter k, we show a significantly better algorithm for SWITCH-FEW-EDGES in Subsection 4.3. We begin by showing the NP-completeness of the problems when k is a part of the input. This is done in Subsection 4.1.

4.1 NP-Completeness

In this subsection, we prove the NP-completeness of SWITCH-FEW-EDGES and SWITCH-MANY-EDGES.

Lemma 4.1 *The problems* SWITCH-FEW-EDGES *and* SWITCH-MANY-EDGES *are polynomially equivalent.*

Proof: Let G be a graph. We use the fact that $\overline{S(G, A)} = S(\overline{G}, A)$ for any set $A \subseteq V_G$. Thus the graph $S(G, A)$ has at most k edges if and only if its complement $S(\overline{G}, A)$ has at least $\binom{n}{2} - k$ edges. □

Therefore it suffices to show the NP-completeness of SWITCH-FEW-EDGES only. We use a reduction of the following well-known NP-complete problem [GJ79].

SIMPLE-MAX-CUT
Input: A graph G, an integer j
Question: Does there exist a partition V_1, V_2 of V_G such that the cut between V_1 and V_2 in G contains at least j edges?

Theorem 4.2 SWITCH-FEW-EDGES *is NP-complete.*

Proof: From an instance (G, j) of SIMPLE-MAX-CUT we create an instance (G', k) of SWITCH-FEW-EDGES in the following way. For each vertex of G we create a corresponding non-adjacent vertex pair in G'. An edge in G is represented by four edges completely interconnecting the two pairs, and a non-edge in G is represented by only two edges connecting the two pairs in a parallel way. More formally, we set

$$\begin{aligned} V_{G'} &= \{v', v'' : v \in V_G\} \\ E_{G'} &= \{\{u', v'\}, \{u'', v''\} : u, v \in V_G, u \neq v\} \cup \{\{u', v''\}, \{u'', v'\} : \{u, v\} \in E_G\}. \end{aligned}$$

The following lemma relates cuts in G to switches of G'.

Lemma 4.3 *The following statements are equivalent:*

(a) There is a cut in G having at least j edges,

(b) there exists a set $A \subseteq V_{G'}$ such that $S(G', A)$ contains at most $2\binom{|V_G|}{2} + 2|E_G| - 4j$ edges.

Proof: For a cut C in G we define a corresponding vertex subset $A = A(C)$ of G'. Suppose that C separates a vertex set V_1 from the remaining vertices of G. We set A to be the set $\{v', v'' : v \in V_1\}$. Note that such a set A satisfies the following condition of legality.

Definition 4.4 *Let A be a vertex subset of G'. We say that a vertex $u \in V_G$ is* broken in A *if A contains exactly one vertex of u', u'', otherwise, we say that u is* sound in A. *We say that a vertex-pair (edge or non-edge) $\{u, v\} \subseteq V_G$ is* broken in A *if at least one of its vertices u, v is broken. Otherwise, we say that $\{u, v\}$ is* sound in A.

If all vertices of G are sound in A, we say that A is legal.

Legality is a desired property, because there is an obvious correspondence between legal sets and cuts in G. Also, the number of edges in $S(G', A)$ is determined by the size of the cut C. The original graph G' contains two edges per every vertex-pair of G and two more edges per every edge in G, which is $2\binom{|V_G|}{2} + 2|E_G|$ edges altogether. Since A is legal, it can easily be checked that every non-edge $\{u, v\}$ in G corresponds to two edges in both G' and $S(G', A)$, regardless of the cut C. For every edge $\{u, v\}$ in the cut C we have u', $u'' \in A$ and v', $v'' \notin A$ (or vice versa), so switching A destroys all the four corresponding edges and creates none. For an edge not present in the cut C, switching A does not modify the corresponding edges, so there are still four of them in $S(G', A)$. To sum it up, $S(G', A)$ has

$$2\binom{|V_G|}{2} + 2|E_G| - 4|C|$$

edges. This proves that the statement (a) implies (b). As for the other implication, by reverting the construction of $A(C)$ from a cut C, we get that it holds for legal sets A. It remains to deal with possible illegal switches.

Lemma 4.5 *For every illegal set A there is a legal set A' such that $S(G', A')$ contains at most the same number of edges as $S(G', A)$.*

Proof: Let A be an illegal set. As can be seen in Fig. 2, a broken non-edge never decreases the resulting number of edges, and thus is no more profitable than a sound non-edge. Therefore, if all broken vertex-pairs are *non-edges*, we can legalize A by removing all vertices of G' that correspond to broken vertices

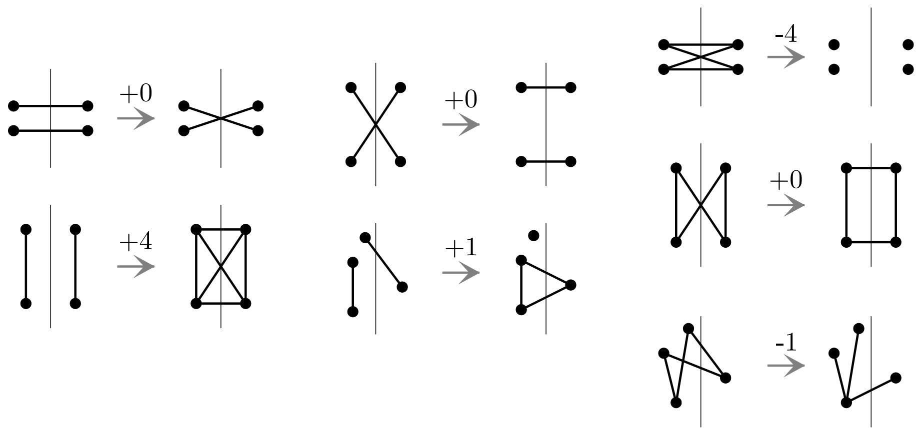

Fig. 2: The possible legal and illegal switches of non-edges (on the left, in the middle) and edges (on the right) in G and their influence on the number of edges in G'.

of G. After this modification, A is legal and it yields a switch with at most the same number of edges as before.

Assume that there are m broken *edges*, where $m > 0$. As shown in Fig. 2, a broken edge could in certain cases decrease the number of edges in G' by more than a sound edge not present in the cut would. We create a legal set A' from A using the following greedy algorithm.

1. $A' := A$.
2. Find a vertex v broken in A'. If there is no such, STOP.
3. Look for sound neighbors of v in G. If there are more such neighbors u having $u', u'' \in A'$ than those having $u', u'' \notin A'$, then set $A' := A' \setminus \{v', v''\}$, otherwise set $A' := A' \cup \{v', v''\}$.
4. Go back to step 2.

It remains to prove that the algorithm finds a legal set that is better than A.

In each iteration of step 3, the algorithm makes one vertex sound. It clearly finishes after a finite number of steps and creates a legal set. It does not modify sound vertices nor sound edges, therefore the number of edges in $S(G', A')$ corresponding to sound vertex-pairs remains unchanged. Also, making a non-edge sound does not increase the number of edges in $S(G', A')$ in comparison to $S(G', A)$.

As we already know, any legal set gives us a cut in G; consider the cut obtained from A'. In each iteration of Step 3, at least half of the involved edges of G became cut edges. Since during the run of the algorithm, each broken edge was made sound exactly once, we know that at least $m/2$ broken edges became cut edges (and decreased the edge number of $S(G', A')$ by at least 3), and at most $m/2$ broken edges became out of the cut (and increased the edge number of $S(G', A')$ by at most 1). Therefore, the edge number in $S(G', A')$ is lower by at least $3m/2 - m/2 = m$, which we assumed to be positive. □

To finish the proof of Lemma 4.3, it remains to prove that (b) implies (a). Let $S(G', A)$ be a switch with at most $2\binom{|V_G|}{2} + 2|E_G| - 4j$ edges. If A is illegal, Lemma 4.5 assures that there is a legal set A' such that

$S(G', A')$ has even less edges. The legal set yields a partition $V_1 = \{v : v', v'' \in A'\}$ and $V_2 = V_G \setminus V_1$ such that the cut between V_1 and V_2 contains at least j edges. □

According to Lemma 4.3, the graph G contains a cut of size at least j if and only if G' is switching-equivalent to a graph with at most $k = 2\binom{|V_G|}{2} + 2|E_G| - 4j$ edges. The size of (G', k) is surely polynomial in the size of (G, j). That concludes the proof of NP-hardness of SWITCH-FEW-EDGES. To prove that SWITCH-FEW-EDGES is in NP, it suffices to note that the positive answer can be certified by a vertex set A that gives us a switch with the desired number of edges. □

4.2 A Connection to Maximum Likelihood Decoding

In this section, we explain how the problem SWITCH-FEW-EDGES may be translated into the language of coding theory, and how it is related to a well-known problem called MAXIMUM LIKELIHOOD DECODING. We derive a consequence of Theorem 4.2.

Let V be a fixed set of n vertices. For an edge set $E \subseteq \binom{V}{2}$, by χ_E we denote the characteristic vector of E, i. e., the element of $\mathbb{Z}_2^{\binom{n}{2}}$ such that $\chi_E(e) = 1$ if and only if $e \in E$. Thus any graph on the vertex set V can be represented by a vector of length $\binom{n}{2}$. The following observation expresses how switching works by means of characteristic vectors.

Observation 4.6 *Let $K_n = (V, \binom{V}{2})$ be the complete graph, let V_1, V_2 be a partition of V and let $S = \{\{x, y\}, x \in V_1,\ y \in V_2\}$ be the corresponding cut in K_n. Then for any $G = (V, E)$,*

$$\chi_{S(G,V_1)} = \chi_{S(G,V_2)} = \chi_E + \chi_S.$$

(Note that the summation is done over $\mathbb{Z}_2$.) Therefore, if we seek a switch of G with the minimum number of edges, we seek a characteristic vector $\chi_E + \chi_S$ with the minimum Hamming weight. Or, equivalently, we seek a cut S in K_n with the minimum Hamming distance between χ_S and χ_E.

It is a well-known fact that the cut space $\mathcal{C}^*(G)$ of a graph G is a vector space, and the cycle space $\mathcal{C}(G)$ is also a vector space orthogonal to $\mathcal{C}^*(G)$. The dimension of $\mathcal{C}^*(G)$ is $|V| - 1$, and $\mathcal{C}^*(G)$ can also be viewed as a linear $[|E|, |V| - 1]$ code with a *parity-check matrix* C whose rows are $|E| - |V| + 1$ linearly independent characteristic vectors of cycles in G. Such a code is called a *graph theoretic code*; the concept of graph theoretic codes has been introduced by Hakimi and Frank [HF65].

The problem of finding a codeword in a linear code that is closest to a given vector is an important problem in coding theory. It can be formulated as a decision problem in the following way.

Problem: MAXIMUM LIKELIHOOD DECODING
Input: A binary $p \times q$ matrix H, a vector $r \in \mathbb{Z}_2^p$, and an integer $w > 0$.
Question: Is there a vector $e \in \mathbb{Z}_2^q$ of Hamming weight at most w such that $He = r$?

This problem was proven to be NP-complete by Berlekamp et al. [BMvT78]. Note that SWITCH-FEW-EDGES is indeed its special case, which we formalize in the following lemma.

Lemma 4.7 SWITCH-FEW-EDGES *is a special case of* MAXIMUM LIKELIHOOD DECODING*, where H is the parity check matrix of the code of cuts in a complete graph.*

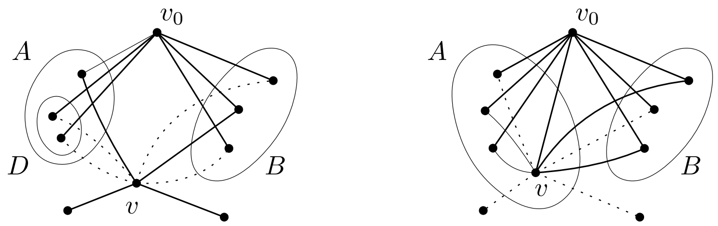

Fig. 3: The graph H in the proof of Lemma 4.8 (left) and Lemma 4.9 (right).

Proof: Having a graph G with edge set E, we set $H = C(K_n)$ (the parity-check matrix of $\mathcal{C}^*(K_n)$), $r = H\chi_E$, and $w = k$. Then a vector e is a solution of MAXIMUM LIKELIHOOD DECODING if and only if $H(e + \chi_E) = \mathbf{0}$, which means that the vector $e + \chi_E$ is an element of the cut space $\mathcal{C}^*(K_n)$ and its Hamming distance from χ_E is at most k.

Therefore, by Observation 4.6 there exists a switch of $S(G, A)$ whose characteristic vector is e. Since the Hamming weight of e is at most k, the switch $S(G, A)$ has at most k edges and we are done. (i) □

Special cases of MAXIMUM LIKELIHOOD DECODING have been studied. It is known that the problem is NP-complete even if we allow unbounded time for preprocessing the code H. This was proven by Bruck and Naor [BN90] by showing that MAXIMUM LIKELIHOOD DECODING is NP-complete for the cut code of a special fixed graph, and therefore no preprocessing can help because this fixed code can be known in advance. Our proof in Subsection 4.1 provides an alternative proof of Bruck and Naor's result by using K_n as the fixed graph.

4.3 Fixed Parameter Tractability

As we have seen in Subsection 4.1, the problem SWITCH-FEW-EDGES is NP-complete. But since the desired graph has at most k edges, each vertex has degree at most k. So the problem is fixed-parameter tractable by similar arguments to the ones used for SWITCH-k-SMALL-DEGS problem. In the remainder of this subsection, we present a much more efficient algorithm for this problem.

The algorithm works in a very similar way as the algorithm from Subsection 3.2, hence we omit some parts of it. We again suppose, that our graph has an isolated vertex v_0 that is not switched in the sought solution. Again, we switch vertices one by one. We denote by l the upper bound for the number of vertices we can still switch and by B the set of vertices already switched. At the beginning $B := \emptyset$ and during the entire algorithm we set $l := k - |B|$. We introduce two rules. They are more powerful and the proofs of their correctness are different.

Lemma 4.8 (Rule 1) *A vertex $v \in (V \setminus B) \setminus \{v_0\}$ of degree $deg_{S(G,B)}(v) > l$ must be switched in order to obtain a solution.*

Proof: Suppose that $v \in (V \setminus B) \setminus \{v_0\}$ is a vertex of degree $d = deg_{S(G,B)}(v) > l$ and $A \subseteq (V \setminus B) \setminus \{v_0\}$ is a set, such that its switching in $S(G, B)$ gives a graph H. We show that if $v \notin A$, then the graph H

(i) We remark that if we consider a gain graph with labels from the group $\mathbb{Z}_2$ and an arbitrary underlying graph G, an analogous reasoning yields a generalized result. If we want to switch the gain graph to contain at most k ones, this is also a special case of MAXIMUM LIKELIHOOD DECODING with H being the parity check matrix of the cut code of G. This, however, is out of scope of this paper.

has too many edges. Denote $D = A \cap N_{S(G,B)}(v)$ (see Fig. 3). Then $\{\{v_0, x\} | x \in D \cup B\}$ and $\{\{v, x\} | x \in N_{S(G,B)}(v) \setminus D\}$ are two disjoint sets of edges of the graph H such that the size of their union is $|B| + |D| + |N_{S(G,B)}(v) \setminus D| = |B| + |N_{S(G,B)}(v)| = d + k - l > k$. Thus the graph H has more than k edges and the switching of the set A does not lead to a solution. □

Lemma 4.9 (Rule 2) *Vertex $v \in V \setminus B \setminus \{v_0\}$ of degree $deg_{S(G,B)}(v) < n - l - 1$ cannot be switched in order to obtain a solution.*

Proof: Let again $d = deg_{S(G,B)}(v) < n - l - 1$, $A \subseteq (V \setminus B) \setminus \{v_0\}$ and $H = S(G, A \cup B)$. Suppose $v \in A$. Then $\{\{v_0, x\} | x \in A \cup B \setminus \{v\}\}$ and $\{\{v, x\} | x \in (V \setminus N_{S(G,B)}(v)) \setminus A\}$ are two disjoint sets of edges of the graph H (see Fig. 3). The size of them together is at least $(|B| + |A| - 1) + (n - d - |A|) = k - l - 1 + n - d > k - l + n - (n - l - 1) - 1 = k$. Thus the graph H has more than k edges and the switching of the set A does not lead to a solution. □

The following boundary lemma is an easy corollary of two previous Lemmas.

Lemma 4.10 (Boundary) *If $n > 2l + 1$, then on each vertex either Rule 1 or Rule 2 applies.*

The rest of the algorithm remains almost the same as in Subsection 3.2, only the kernel bound is $2k$ vertices, hence we omit the overview of the algorithm here. Again if $n = 2p + 2$ or $n = 2p + 1$ for some $0 \le p \le k - 1$, then after switching $k - p$ vertices we can decide about each vertex whether it should be switched or not according to Lemma 4.10. Thus we have at most $\binom{2p+1}{k-p} \cdot k$ candidate sets for $p \ge k/2$ and at most $2^{2p+1} \le 2^k$ candidate sets for $p < k/2$. To count the time complexity, we must search for the biggest number among

$$\binom{2\lceil k/2\rceil + 1}{\lfloor k/2\rfloor}, \binom{2\lceil k/2\rceil + 3}{\lfloor k/2\rfloor - 1}, \ldots, \binom{2p+1}{k-p}, \ldots, \binom{2k-3}{2}, \binom{2k-1}{1},$$

since for $k \ge 2$ it holds that $2^k \le k \cdot \binom{2\lceil k/2\rceil+1}{\lfloor k/2\rfloor}$.

By comparing again two neighboring terms $\binom{2p+1}{k-p}$ and $\binom{2p+3}{k-p-1}$ we get the cubic equation $(3p - k + 4) \cdot (3p - k + 3) \cdot (3p - k + 2) = (2p + 3) \cdot (2p + 2) \cdot (k - p)$, with parameter k. The equation has again only one real root

$$\begin{aligned} p_0 = {} & \frac{1}{93} \cdot (31k - 91 + (3844k^3 + 11904k^2 + 9579k + 1403 + \\ + {} & 93\sqrt{3}\sqrt{496k^6 + 2976k^5 + 6088k^4 + 3888k^3 - 3541k^2 - 6848k - 3900})^{1/3} \\ + {} & (124k^2 + 310k + 469) \cdot (3844k^3 + 11904k^2 + 9579k + 1403 + \\ + {} & 93\sqrt{3}\sqrt{496k^6 + 2976k^5 + 6088k^4 + 3888k^3 - 3541k^2 - 6848k - 3900})^{-1/3}), \end{aligned}$$

which is $p_0 = 0.611492k + o(k)$. Hence, for k large enough, $\binom{\lceil 1.2230k\rceil+1}{\lfloor 0.3885k\rfloor}$ is the biggest term.

Again using standard factorial approximation we have:

$$\begin{aligned} & \binom{\lceil 1.2230k\rceil + 1}{\lfloor 0.3885k\rfloor} \le \binom{1.2231k}{0.3885k} = \frac{1.2231k}{e} \frac{(1.2231)^{1.2231k}}{(0.3885)^{0.3885k} \cdot (0.8346)^{0.8346k}} = \\ = {} & \frac{1.2231k}{e} exp(k(1.2231 \ln 1.2231 - 0.3885 \ln 0.3885 - 0.8346 \ln 0.8346)) \le \\ \le {} & 2.1479^k, \end{aligned}$$

for k large enough.

Our results are summarized by the following theorem:

Theorem 4.11 SWITCH-FEW-EDGES *is fixed-parameter tractable and there is an algorithm running in time* $\mathcal{O}\left(2.148^k \cdot n + m\right)$.

5 Searching for a Switch with a cn-Clique

In this section, we consider the following problem. We assume that c is a fixed constant from the interval $(0, 1)$.

SWITCH-cn-CLIQUE
Input: Graph $G = (V, E)$ on n vertices
Question: Is there a subset of vertices $A \subseteq V$ such that the graph $S(G, A)$ contains a clique of size at least cn?

Theorem 5.1 *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. It is clear that the problem is in NP – a polynomial-size certificate contains vertex subsets A and C such that $S(G, A)[C]$ is a clique of the desired size. In the case of irrational c, we assume that an oracle can be used giving for any n the value of $\lceil cn \rceil$ in constant time. This ensures that the certificate can be checked in time polynomial in n.

Each step is a reduction of the SAT problem which is known to be NP-complete [GJ79].

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 $k \geq 2$.

Let $G = G_{p,q}(\varphi)$ be a graph constructed in the way illustrated in Figure 4. 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 to nothing else.
- 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 $l_2 = \neg l_1$.

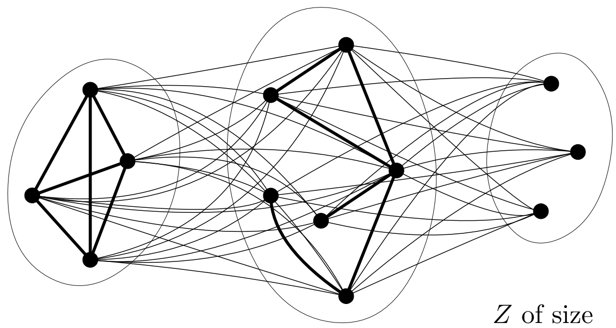

Fig. 4: The graph $G_{p,q}(\varphi)$.

Lemma 5.2 *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 5.2 immediately gives us the following corollary.

Corollary 5.3 *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 5.4 *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 $A \subseteq V_G$ such 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 $l > 2$. Moreover, $p \geq 1$, 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.

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. $\square$

Corollary 5.3 and Lemma 5.4 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-complete for every rational constant $c \in (0, 1)$.

To prove the NP-completeness of SWITCH-cn-CLIQUE for irrational numbers c as well, we use a consequence of the PCP theorem of Arora et al. [ALM+98] (stated analogously to the following statement in [Vaz01, Theorem 29.7]).

Theorem 5.5 *There is a constant $\varepsilon > 0$ for which there is a gap-introducing reduction from SAT to MAX-3SAT that transforms a Boolean formula ψ to $T(\psi)$ 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 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 5.6. Let $\{x\}$ denote the fractional part of x. Lemma 5.6 is a special case of the (one-dimensional) Kronecker Theorem which states that for every irrational number α the sequence $\{n\alpha\}$, where $n = 1, 2, \ldots$, is dense in the interval $[0, 1]$ (see [HW08]; the Theorem was first published in 1884).

Lemma 5.6 *For any irrational number $\alpha \in [0, 1]$, any $\varepsilon > 0$ and $r \in \mathbb{R}$ there exists $n \in \mathbb{N}$ such that $n > r$ and $\{n\alpha\} < \varepsilon$.*

For our purposes we need the following statement:

Lemma 5.7 *For each irrational $c \in (0, 1)$, each $\varepsilon > 0$ and $m > 0$, there exist p, $q \in \mathbb{N}$ such that $\frac{p}{q} \in (0, 1)$, $p > m$ 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. Then we check that the obtained numbers p and q satisfy the conditions of the Lemma. In order for the interval $(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c})$ to contain an integer, p must satisfy the condition

$$\left\{\frac{p}{c}\right\} < \frac{\varepsilon}{4c}. \tag{1}$$

We also want that $p > m$ (which is given in the statement of the Lemma), and additionally we request that the following condition holds (which will prove useful later):

$$p > \frac{\varepsilon}{4(1-c)}. \tag{2}$$

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 5.6 for $\alpha = \{\frac{1}{c}\}$, $\varepsilon' = \frac{\varepsilon}{4c}$ and $r = \max(m, \frac{\varepsilon}{4(1-c)})$ 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 $\left(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c}\right)$. 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 (1) together give us the other inequality $q > \frac{p}{c} - \frac{\varepsilon}{4c}$.

Moreover, from (2) we obtain

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

so any integer q in the interval $\left(\frac{p}{c} - \frac{\varepsilon}{4c}, \frac{p}{c}\right)$ is larger than p. In particular, q is a positive integer, and $\frac{p}{q} \in (0, 1)$. 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 5.5, and p, q the integers given by Lemma 5.7 for ε and c and such that $p > 2$. We take an instance ψ of 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, and n stands for the number of vertices of G.

If ψ is satisfiable, we have again by Corollary 5.3 that $G[L]$ contains a k-clique, and by Lemma 5.4 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 5.8 *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 $A \subseteq V_G$ the graph $S(G, A)$ contains a clique C of size larger than $pl + p - \varepsilon k$. Then (similarly as in the proof of Lemma 5.4) 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, 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.

We have requested that $p > 2$. Then $l < (p - 1)l + p < pl + p - \varepsilon k$. 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. By Theorem 5.5 and Lemma 5.2, the maximum clique size in $G[L]$ is $(1 - \varepsilon)k$, which is a contradiction. □

By Lemma 5.8, 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-complete. $\square$

6 Parameterized Complexity of the Embedding Problem

In this section we examine the following problem:

SWITCH-EMBED **Input:** Graphs G, H **Parameter:** $\lvert V(H)\rvert$ **Question:** Is there a graph that is switching-equivalent to the input graph G and contains the input graph H as an induced subgraph?

It has been shown in [EHHR00a] that it is NP-Complete to decide whether a switching class contains a graph with a clique of size at least k. There is also an easy parameterized reduction from CLIQUE showing the problem SWITCH-EMBED is W[1]-hard even if we restrict the input graph H to be complete graph.

Proposition 6.1 *It is W[1]-hard to decide whether there is a graph that is switching-equivalent to the input graph and contains a clique of size k, k being the parameter of the problem.*

Proof: We reduce the CLIQUE problem that is well known to be W[1]-complete [DF99]. Let (G,k) be an instance of CLIQUE. Consider a graph G' obtained from G by adding a clique K of size k and connecting all its vertices with all the vertices of G. Then the switching class of G' contains a graph with a clique of size $2k$ if and only if G' contains such a clique, which is if and only if G contains a clique of size k.

To see this, first assume that some vertices of K are contained in a $2k$-clique C in a switch $S(G',A)$ of G'. Since the vertices of C are all pairwise adjacent in $S(G',A)$, and the vertices of K are adjacent to all vertices of G in G', the vertices of C are all contained either in A or in $V(G') \setminus A$. Hence, either A or $V(G') \setminus A$ contains the clique C both in $S(G', A)$ and in G', thus G' also contains a clique of size $2k$.

If the clique C consists solely of the vertices of G, then one of the sets $A, V(G) \setminus A$ contains at least half of C, and thus there is a clique of size k in G. □

The problem can be also easily shown to be in W[1].

Proposition 6.2 *The problem* SWITCH-EMBED *is in W[1].*

For the proof we use the characterization of W[1] by Nondeterministic Random Access Machines as proposed in [CFG03].

A nondeterministic random access machine (NRAM) model is based on the standard deterministic random access machine (RAM) model. A single nondeterministic instruction "GUESS" is added, whose semantics is: *Guess a natural number less than or equal to the number stored in the accumulator and store it in the accumulator.* Acceptance of an input by an NRAM is defined as usually for nondeterministic machines. The steps of computation of an NRAM that execute a GUESS instruction are called *nondeterministic steps*.

Definition 6.3 *An NRAM program $\mathcal{P}$ is* tail-nondeterministic k-restricted *if there are computable functions f and g and a polynomial p such that on every run with input $(x, k) \in \Sigma^* \times \mathbb{N}$ the program $\mathcal{P}$*

- *performs at most $f(k) \cdot p(n)$ steps;*
- *uses at most the first $f(k) \cdot p(n)$ registers;*
- *contains numbers $\leq f(k) \cdot p(n)$ in any register at any time;*

and all nondeterministic steps are among the last $g(k)$ steps of the computation. Here $n = |x|$.

The following characterization is crucial for our proof:

Theorem 6.4 (Flum, Grohe [CFG03]) *A parameterized problem P is in W[1] if and only if there is a tail-nondeterministic k-restricted NRAM program deciding P.*

Proof of Proposition 6.2: For the proof we introduce the program `SwitchEmbed` that takes graphs G and H as an input, and there is an accepting computation of `SwitchEmbed` on G and H if and only if there is a switch of G containing H as an induced subgraph. We present it in a higher level language that can be easily translated to the NRAM instructions. The program is obviously tail-nondeterministic $|H|$-restricted. It can easily be modified to handle the non-induced case.

Program `SwitchEmbed`(G, H)

1. Denote $k := |H|, V(H) = \{h_1, \ldots, h_k\}$;
2. Guess k distinct vertices $v_1, \ldots, v_k$ from $V(G)$;
3. Denote $S := \{v_1, \ldots, v_k\}$;
4. Guess $A \subseteq S$;
5. Denote $f : S \to V(H), v_i \mapsto h_i (\forall i, 1 \leq i \leq k)$
6. If f is a graph isomorphism between $S(A, G[S])$ and H then ACCEPT; else REJECT;

□

7 Open Problems

To conclude, we mention some open problems related to the problems we have solved in this paper.

Switching to a k-degenerated graph An $\mathcal{O}\left(n^{k+3}\right)$-time algorithm for this problem follows from the results of Ehrenfeucht et al. [EHHR00a]. If k is a part of the input, no polynomial time algorithm is known. The problem is also not known to be NP-complete. Fixed-parameter tractability would be a good alternative.

Parameterization above guaranteed value It is not known whether it is possible to decide in polynomial time whether the input graph is switching-equivalent to a graph having all degrees at least $\lceil n/2 \rceil + k$, where k is a fixed constant.

Swiching to an H-free graph The question whether the input graph can be switched to a graph not containing the graph H (which is also part of the input) is not even known to be in NP, it only trivially falls into the class Σ_2^P of the polynomial hierarchy. It is known to be in P for the following fixed graphs H (and their complements): for K_2 (this can be simply proved, see also [HHW02]), for $K_{1,2}$ ([KNZ92]), for K_3 ([Hay96, HHW02]), P_4 ([Her99]), and $K_{1,3}$ ([JK08]). On the contrary, there are infinitely many graphs H in a certain class of graphs called "hedgehogs" for which the problem is NP-complete [Jel11]. However, these are the only few exceptions for which the complexity has been determined. It is not even known whether the problem is either polynomial or NP-complete for each fixed graph H.

Acknowledgements

The authors would like to thank Jiří Sgall for useful suggestions, Martin Klazar for skillful advice, and the anonymous referees for helping to improve the paper.

References

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of ACM*, 45(3):501–555, 1998. An extended abstract appeared in FOCS 1992.

[BMvT78] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, IT-24:384–386, 1978.

[BN90] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Trans. Inform. Theory*, 36:381–384, 1990.

[CC80] C. J. Colbourn and D. G. Corneil. On deciding switching equivalence of graphs. *Discrete Appl. Math*, 2:181–184, 1980.

[CFG03] Y. Chen, J. Flum, and M. Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *IEEE Conference on Computational Complexity*, pages 13–29. IEEE Computer Society, 2003.

[DF99] R. G. Downey and M. R. Fellows. *Parametrized Complexity*. Monographs in Computer Science. Springer, 1999.

[EHHR00a] A. Ehrenfeucht, J. Hage, T. Harju, and G. Rozenberg. Complexity issues in switching of graphs. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Theory and Application to Graph Transformations*, volume 1764 of *LNCS*, pages 59–70. Springer, Heidelberg, 2000.

[EHHR00b] A. Ehrenfeucht, J. Hage, T. Harju, and G. Rozenberg. Pancyclicity in switching classes. *Inf. Process. Lett.*, 73(5-6):153–156, 2000.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[Hag01] J. Hage. *Structural Aspects Of Switching Classes*. PhD thesis, Leiden Institute of Advanced Computer Science, 2001. http://people.cs.uu.nl/jur/2s.html.

[Hay96] R. B. Hayward. Recognizing P_3-structure: A switching approach. *J. Comb. Theory, Ser. B*, 66(2):247–262, 1996.

[Her99] A. Hertz. On perfect switching classes. *Discrete Applied Mathematics*, 94(1-3):3–7, 1999.

[HF65] S. L. Hakimi and H. Frank. Cut-set matrices and linear codes. *IEEE Trans. Inform. Theory*, IT-11:457–458, 1965.

[HHW02] J. Hage, T. Harju, and E. Welzl. Euler graphs, triangle-free graphs and bipartite graphs in switching classes. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proceedings of ICGT 2002*, volume 2505 of *LNCS*, pages 148–160. Springer, Heidelberg, 2002.

[HW08] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 6th edition, 2008.

[Jel08] E. Jelínková. Three NP-complete optimization problems in seidel's switching. In L. Brankovich, Y. Lin, and W. F. Smyth, editors, *Proceedings of IWOCA 2007*, pages 121–135. College Publications, 2008.

[Jel11] E. Jelínková. Switching to hedgehog-free graphs is NP-complete. In M. Ogihara and J. Tarui, editors, *TAMC 2011*, volume 6648 of *LNCS*, pages 463–470. Springer, Heidelberg, 2011.

[JK08] E. Jelínková and J. Kratochvíl. On switching to H-free graphs. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *ICGT 2008*, volume 5214 of *LNCS*, pages 379–395. Springer, Heidelberg, 2008.

[KNZ92] J. Kratochvíl, J. Nešetřil, and O. Zýka. On the computational complexity of Seidel's switching. In *Combinatorics, graphs and complexity, Proc. 4th Czech. Symp., Prachatice 1990*, volume 51 of *Annals of Discrete Math.*, pages 161–166. North Holland, Amsterdam, 1992.

[Kra03] J. Kratochvíl. Complexity of hypergraph coloring and Seidel's switching. In Hans L. Bodlaender, editor, *WG 2003*, volume 2880 of *LNCS*, pages 297–308. Springer Verlag, 2003.

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[Sei74] J. J. Seidel. Graphs and two-graphs. In *Proc. 5th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, Winnipeg, Canada, 1974. Utilitas Mathematica Publishing Inc.

[Sei76] J. J. Seidel. A survey of two-graphs. In *Proc. Colloquio Internazionale sulle Teorie Combinatorie (Rome, 1973)*, volume 17, pages 481–511, Rome, 1976. Accademia Nazionale dei Lincei.

[ST81] J. J. Seidel and D. E. Taylor. Two-graphs, a second survey. In L. Lovász and V. T. Sós, editors, *Algebraic Methods in Graph Theory (Proc. Internat. Colloq., Szeged, 1978)*, volume II, pages 689–711, Amsterdam, 1981. North-Holland.

[Suc08] O. Suchý. Some parametrized problems related to Seidel's switching. In L. Brankovich, Y. Lin, and W. F. Smyth, editors, *Proceedings of IWOCA 2007*, pages 148–157. College Publications, 2008.

[Vaz01] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.