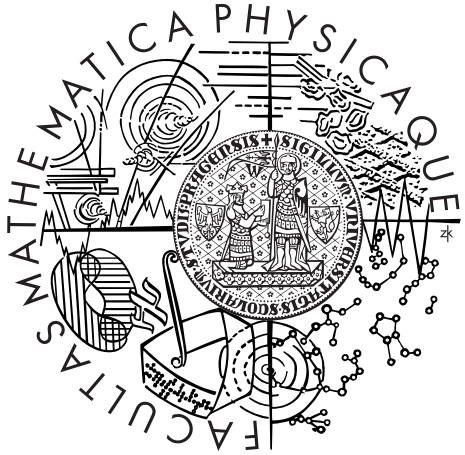
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Eva Jelínková

Computational Complexity in Graph Theory

Department of Applied Mathematics

Supervisor of the doctoral thesis: Prof. RNDr. Jan Kratochvíl, CSc.

Study programme: Computer Science

Study branch: 4I4 - Discrete Models and Algorithms

Prague 2016

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague, April 13, 2016

Abstracts

Title: Computational Complexity in Graph Theory

Author: Eva Jelínková

Department: Department of Applied Mathematics

Supervisor: Prof. RNDr. Jan Kratochvíl, CSc., Department of Applied Mathematics

Abstract: We address problems from graph theory, especially from the computational complexity point of view. In the first part of the thesis we address the computational complexity of problems related to Seidel's switching of graphs. We prove that the problem to decide if a given graph can be switched to contain at most a given number of edges is NP-complete, even for graphs with bounded density. We thus partially answer a question of Matoušek and Wagner [Discrete Comput. Geom. 52, no. 1, 2014]. We also describe infinitely many graphs H such that it is NP-complete to decide if a given graph is switching-equivalent to a graph that does not contain H as an induced subgraph. We thus close an open problem of Kratochvíl, Nešetřil and Zýka [Annals of Discrete Math. 51, 1992].

In the second part of the thesis we address the topic of matchings under preferences. We focus on the housing market problem, in particular, on the model with duplicate houses. We present a 2-approximation algorithm for the maximum number of satisfied agents when the preference lists of agents are trichotomic. On the other hand, we prove that the number is NP-hard to approximate within a factor smaller than 21/19, and if the preferences are not required to be trichotomic, then the number is NP-hard to approximate within a factor smaller than 1.2. In addition to these results, we show an asymptotically optimal improvement of the equilibrium algorithm in the setting where agents' preferences are strict linear orderings of house types.

Keywords: Seidel's switching, graph theory, computational complexity, Housing Market problem, economic equilibrium

Název práce: Výpočetní složitost v teorii grafů

Autor: Eva Jelínková

Katedra: Katedra Aplikované Matematiky

Vedoucí disertační práce: Prof. RNDr. Jan Kratochvíl, CSc., Katedra Aplikované Matematiky

Abstrakt: Zabýváme se problémy teorie grafů, zejména z hlediska výpočetní složitosti. V první části práce se věnujeme výpočetní složitosti problémů souvisejících se Seidelovým přepnutím grafů. Uvažujeme rozhodující problém, zda daný graf lze přepnout tak, aby obsahoval nejvýše daný počet hran. Dokážeme, že tento problém je NP-úplný, dokonce i pro grafy s omezenou hustotou. Částečně tak odpovídáme na otázku Matouška a Wagnera [Discrete Comput. Geom. 52, no. 1, 2014]. Popisujeme také nekonečně mnoho grafů H, pro které je NP-těžké rozhodnout, zda pro daný graf existuje graf, který je s ním ekvivalentní v přepnutí, a zároveň neobsahuje H jako indukovaný podgraf. Tímto řešíme otevřený problém Kratochvíla, Nešetřila a Zýky [Annals of Discrete Math. 51, 1992].

Ve druhé části práce se zabýváme tématem párování s preferencemi. Zaměřujeme se na problém trhu s domy, konkrétně na model s duplicitními domy. Popisujeme 2-aproximační algoritmus pro maximální počet spokojených agentů v případě, kdy preferenční seznamy agentů jsou trichotomické. Zároveň ale dokazujeme, že tento počet je NP-těžké aproximovat s poměrem menším než 21/19, a pokud preferenční seznamy agentů nemusí být trichotomické, potom je těžké tento počet aproximovat s poměrem menším než 1.2. Kromě těchto výsledků také popisujeme asymptoticky optimální vylepšení algoritmu pro ekonomické ekvilibrium pro situaci, kdy preferenční seznamy agentů jsou ostrá lineární uspořádání typů domů.

Klíčová slova: Seidelovo přepnutí, teorie grafů, výpočetní složitost, problém trhu s domy, ekonomické ekvilibrium

I am very grateful to my advisor, prof. Jan Kratochvíl, for his scientific guidance and generous care. I am also very grateful to prof. Katarína Cechlárová for having introduced to me the topic of housing markets and for having spent a lot of time by discussions with me.

Further, I would like to thank the Department of Applied Mathematics for generous support and to my colleagues from there for a very friendly atmosphere. I would also like to thank the Icelandic Centre of Excellence in Theoretical Computer Science for generously hosting me while my husband was staying at Reykjavik University. I am grateful to all people whom I met there for scientific discussions and work on joint papers; I would especially like to thank Magnús Halldórsson for discussions related to some results contained in this thesis.

My special thanks go to my husband Vítek for scientific advice as well as great and patient support that enabled me to devote a lot of time to research and work on this thesis. I would also like to thank all in my family and friends who supported me in any way, also by their prayers.

Contents

List of My Research Papers

This thesis is based on the following papers:

- Katarína Cechlárová and Eva Jelínková. An efficient implementation of the equilibrium algorithm for housing markets with duplicate houses. *Information Processing Letters*, 111(13):667–670, 2011.

  - Katarína Cechlárová and Eva Jelínková. Approximability of economic equilibrium for housing markets with duplicate houses. In Petr Kolman and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science*, volume 6986 of *Lecture Notes in Computer Science*, pages 95–106. Springer, Heidelberg, 2011.

- Eva Jelínková and Jan Kratochvíl. On switching to H-free graphs. *Journal of Graph Theory*, 75(4):387–405, 2014.

  Parts of the paper appeared in the conference papers:

  - Eva Jelínková and Jan 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.

  - Eva 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.

- Eva Jelínková, Ondřej Suchý, Petr Hliněný, and Jan Kratochvíl. Parameterized problems related to Seidel's switching. *Discrete Mathematics and Theoretical Computer Science*, 13(2):19–42, 2011.

  Parts of the paper appeared in the conference paper:

  - Eva Jelínková. Three NP-complete optimization problems in Seidel's switching. In L. Brankovich, Y. Lin, and W. F. Smyth, editors, *Proceedings of the International Workshop on Combinatorial Algorithms 2007*, pages 121–135. College Publications, 2008.

  - Vít Jelínek, Eva Jelínková, and Jan Kratochvíl. On the hardness of switching to a small number of edges. *Submitted.*

My other papers, not included in the thesis:

- Alexander Burstein, Vít Jelínek, Eva Jelínková, and Einar Steingrímsson. The Möbius function of separable and decomposable permutations. *Journal of Combinatorial Theory, Series A*, 118(8):2346–2364, 2011.

  Parts of the paper appeared in the conference paper:

  - Vít Jelínek, Eva Jelínková, and Einar Steingrímsson. The Möbius function of separable permutations (extended abstract). In *22nd International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2010)*, DMTCS Proceedings, pages 773–784. 2010.

- Anders Claesson, Vít Jelínek, Eva Jelínková, and Sergey Kitaev. Pattern avoidance in partial permutations. *Electronic Journal of Combinatorics*, 18:#P25, 2011.

  Parts of the paper appeared in the conference paper:

  - Anders Claesson, Vít Jelínek, Eva Jelínková, and Sergey Kitaev. Pattern avoidance in partial permutations (extended abstract). In *22nd International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2010)*, DMTCS Proceedings, pages 625–636. 2010.

  - Vít Jelínek, Eva Jelínková, Jan Kratochvíl, and Bernard Lidický. Clustered planarity: embedded clustered graphs with two-component clusters. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 121–132. Springer, Heidelberg, 2009.

- Vít Jelínek, Eva Jelínková, Jan Kratochvíl, Bernard Lidický, Marek Tesař, and Tomáš Vyskočil. The planar slope number of planar partial 3-trees of bounded degree. *Graphs and Combinatorics*, 29(4):981–1005, 2013.

  Parts of the paper appeared in the conference paper:

  - Vít Jelínek, Eva Jelínková, Jan Kratochvíl, Bernard Lidický, Marek Tesař, and Tomáš Vyskočil. The planar slope number of planar partial 3-trees of bounded degree. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 304–315. Springer, Heidelberg, 2010.

- Eva Jelínková, Jan Kára, Jan Kratochvíl, Martin Pergel, Ondřej Suchý, and Tomáš Vyskočil. Clustered planarity: small clusters in Eulerian graphs. *Journal of Graph Algorithms and Applications*, 13(3):379–422, 2009.

  Parts of the paper appeared in the conference paper:

◦ Eva Jelínková, Jan Kára, Jan Kratochvíl, Martin Pergel, Ondřej Suchý, and Tomáš Vyskočil. Clustered planarity: small clusters in Eulerian graphs. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 303–314. Springer, Heidelberg, 2008.

Introduction

Graph theory can be used to model a large scale of natural problems from various fields. Objects and relationships between them can be described by graphs. When translated into the language of graph theory, some problems can be solved by graph algorithms.

Unfortunately, for many problems, a fast enough algorithm is not known. This is studied by the theory of computational complexity which classifies problems according to their "hardness". We focus on the computational complexity of certain problems in graph theory. We mainly study the borderline between NP-complete problems and problems for which there is a polynomial-time algorithm known.

This thesis has two parts. In the first part we deal with problems related to Seidel's switching – 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 one by a sequence of switches.

Seidel's switching 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 [Sei76, Sei74, ST81, Sei92]. There are various algebraical results related to Seidel's switching [Cam77, Wel84, Doo79].

In this thesis, we focus on the computational complexity of problems related to Seidel's switching. This direction has been pursued by Colbourn and Corneil [CC80], Kratochvíl et al. [KNZ92], Ehrenfeucht at al. [EHHR00a], and others. Applications of Seidel's switching can also be found in graph algorithms solving other issues, such as the P_3-structure recognition problem [Hay96] or the recognition of circular permutation graphs [RU82, Sri96].

In Chapter 1 we introduce Seidel's switching in more detail and we also give formal definitions and basic properties of switching.

In Chapter 2 we prove that it is NP-complete to decide if a given graph can be switched to contain at most a given number of edges, even for graphs with bounded density. We thus partially answer a question of Matoušek and Wagner [MW14]. Our results are also related to a problem of Hage [Hag01].

In Chapter 3 we focus on the problem if, for a fixed graph H, a given graph is switching-equivalent to a graph that does not contain H as an induced subgraph. It has been an open problem [KNZ92] whether there exists a graph H such that the problem is NP-complete. We close this problem by giving infinitely many such graphs H. Further, we describe the class of forbidden induced subgraphs

for graphs switching-equivalent to a forest of bounded vertex degrees.

In the second part of the thesis we address the topic of matchings under preferences. The concept of finding a matching that satisfies certain criteria occurs in many practical applications, such as assignment of students to universities, families to housing, kidney transplant patients to donors, children to schools, junior doctors to hospitals.

Such a setting can be modelled by a graph where vertices represent participating agents or units of good (such as a house). Agents may express preferences over the outcomes and the goal is to find an assignment that optimizes the satisfaction of the participants. There are various concepts that define the optimality of the result.

This topic has been subject to a lot of research from the point of view of various areas, such as game theory and economics, computational social choice fields, algorithms and complexity. Practical applications include assigning junior doctors to hospitals in the Japan Residency Matching Program [JRM], in the Canadian Resident Matching Service [CAR] or in the National Resident Matching Program in the U. S. [NRM]; finding kidney exchanges involving incompatible patient-donor pairs [RSÜ04, ABS07, MO12], large residence exchange fairs for subsidized public housing in China [Yua96].

In this thesis, we focus on the housing market model introduced by Shapley and Scarf [SS74]. A housing market consists of a set of agents and a set of houses. Each agent owns one house and has preferences over some of the other houses. The aim of each agent is to get the house he finds the best possible. This model can be applied, for example, in campus housing [SÜ05], assigning students to schools [APR09], and kidney transplantation [RSÜ04].

We focus on the concept of an *economic equilibrium* in the housing market: houses are assigned prices and at these prices, each agent "sells" his house and with the received amount of money he "buys" a house. The set of prices together with the associated trading is called an economic equilibrium if each agent gets the most preferred house that he can afford. We deal with a variant of the housing market model proposed by Fekete et al. [FSW03] where houses are not necessarily unique: some houses have the same *type* and must then have equal price, and agents are indifferent between houses of the same type. This model is called a housing market with *duplicate houses*.

Fekete et al. [FSW03] proved that the problem to decide the existence of an economic equilibrium in a market with duplicate houses is NP-complete [FSW03]. In their setting, there may be *ties* – agents may be indifferent also between houses of different type. Further, Cechlárová and Fleiner [CF10] showed that the problem remains NP-complete even if each agent distinguishes only three classes of house types: desired houses, houses of the same type as his original house and unacceptable houses, and he is indifferent between house types within each class. They call such preferences *trichotomic*. On the other hand, Cechlárová and Fleiner [CF10] proved that if the agents are not allowed to be indifferent between houses of different type, then the economic equilibrium can be found in polynomial time.

In Chapter 4 we give a more thorough introduction into the topic and necessary definitions. In Chapter 5 we show an asymptotically optimal improvement of the equilibrium algorithm of Cechlárová and Fleiner [CF10] in the setting where agents' preferences are strict linear orderings of house types.

In Chapter 6 we focus on the approximability of the maximum number of satisfied agents in housing markets with duplicate houses when preferences may contain ties between different house types. We present a 2-approximation algorithm for the special case when preferences are trichotomic.

On the other hand, we prove that the problem is NP-hard to approximate within a factor smaller than 21/19, even if the preference lists are trichotomic, the maximum number of houses of the same type is two, and agents endowed with a house of the same type have the same preference lists. We further prove that if the preferences are not required to be trichotomic, then the maximum number of satisfied agents is NP-hard to approximate within a factor smaller than 1.2. This complements the results of Cechlárová and Schlotter [CS10] regarding the hardness of computation of the minimum number of unsatisfied agents from the parameterized complexity viewpoint.

Preliminaries

If A is a set and k is an integer, then by $\binom{A}{k}$ we denote the set of all k-element subsets of A. The symmetric difference of sets A and B is denoted by $A \triangle B$. A *partition* of a set A is a family of nonempty subsets of A such that every two of them are disjoint, and their union is A.

A graph G is a pair (V, E), where V (denoted also by $V(G)$) is the set of *vertices of* G and E (denoted also by $E(G)$) is the set of *edges of* G. If there is an edge $\{u, v\}$ in G, we say that the two vertices u and v are *adjacent* or *neighbors*. We also say that u and v are *end-vertices* of the edge $\{u, v\}$, and that they are *incident* to the edge $\{u, v\}$.

A *complement* of a graph G is the graph $\overline{G} = (V, \binom{V}{2} \setminus E)$. A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A graph H is an *induced subgraph* of G if it is a subgraph of G and $E(H) = E(G) \cap \binom{V(H)}{2}$; we also write $H \leq G$ and say that G *contains* H *as an induced subgraph*. If $H \leq G$ but $H \neq G$, we write $H < G$.

If $A \subseteq V(G)$, then we denote by $G[A]$ the graph $(A, E(G) \cap \binom{A}{2})$, and we call it the *subgraph of* G *induced by* A. By $G \setminus A$ we denote the graph $G[V(G) \setminus A]$. If E' is a subset of E, we also define $G \setminus E'$ as the graph $(V, E \setminus E')$.

We say that a partition V_1, V_2 of $V(G)$ is a *cut* of G. For a cut V_1, V_2, the set of edges that have exactly one end-vertex in V_1 is denoted by $\mathrm{cutset}(V_1)$, and the edges of $\mathrm{cutset}(V_1)$ are called *cut-edges*. When there is no danger of confusion, we also say that a single subset $V_1 \subseteq V(G)$ is a cut (meaning the partition V_1, $V(G) \setminus V_1$).

The graph $G = (V, \binom{V}{2})$ is called a *complete graph* and denoted by K_n, where $n = |V|$. A *k-clique* is a complete graph or a complete subgraph on k vertices. A graph G is called *bipartite* if $V(G)$ can be partitioned into two sets such that in $E(G)$, there is no edge between two vertices in the same set. Let U and V be disjoint sets such that $|U| = n$ and $|V| = m$. The graph $(U \cup V, \{\{u, v\} : u \in U, v \in V\})$ is called the *complete bipartite graph* with parts of size n and m, and denoted by $K_{n,m}$. A graph with n vertices and no edges is called a *discrete graph* and denoted by I_n. A vertex subset of a graph G which induces a subgraph with no edges is called an *independent set*.

A *path* with n vertices is a graph formed by distinct vertices $v_1, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$; it is denoted by P_n. A *cycle* with n vertices is denoted by C_n and defined as a graph with distinct vertices $v_1, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$.

A graph G is called *connected* if for every two vertices $u, v \in V(G)$, there is

a path from u to v in G. Maximal connected subgraphs of a graph are called *connected components*. If a graph does not contain a cycle as a subgraph then it is called a *forest* or an *acyclic graph*. A *tree* is a connected forest.

A sequence $v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k$ is called a *walk* in G if for every $i = 1, \ldots, k-1$, the set $E(G)$ contains the edge e_i and $e_i = \{v_i, v_{i+1}\}$. A walk is *closed* if $v_1 = v_k$. Let G be connected; a closed walk in G is called an *Eulerian walk* if it uses each edge of G exactly once. A graph G is called *Eulerian* if each connected component of G has an Eulerian walk.

The *degree* of a vertex v in a graph G is the number of edges of G incident to v, and it is denoted by $\deg_G(v)$. A vertex is called *isolated* if it has degree 0. A graph is *d-regular* if all its vertices have degree d. A *matching* in G is a 1-regular subgraph of G, and a matching in G is called *perfect* if every vertex of G is incident to an edge in the matching. The *neighborhood* of a vertex v in G is the set $N_G(v)$ of vertices adjacent to v.

We also deal with directed graphs. A *directed graph (digraph)* is a pair $D = (V, E)$, where V (denoted also by $V(D)$) is a set of vertices and E (denoted also by $E(D)$) is a set of arcs. We use the notation (u, v) to denote an arc leading from the vertex u to the vertex v. We also say that u is an *in-neighbor* of v and that v is an *out-neighbor* of u. A vertex $v \in V$ is a *sink* if it has no out-neighbors.

An arc (u, u) is called a *loop*. Two arcs (u, v) and (u', v') are called *parallel* if $u = u'$ and $v = v'$. We consider directed graphs that may contain parallel arcs as well as loops. Analogously to undirected graphs, the notions of a *subgraph* (or *subdigraph*) and an *induced subdigraph* are defined.

A *walk* in a digraph $G = (V, E)$ is a sequence $v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k$ such that for every $i = 1, \ldots, k-1$, the set $E(G)$ contains the arc e_i and $e_i = (v_i, v_{i+1})$. We say that a vertex v is a *successor* of a vertex u in G if G contains a walk from u to v.

A digraph $G = (V, E)$ is *strongly connected* if for each pair u, v of vertices in V there is a walk from u to v as well as a walk from v to u in G. A *strongly connected component* (SCC for short) of a digraph G is a maximal strongly connected subdigraph of G. The *condensation* $G^* = (V^*, E^*)$ of a digraph $G = (V, E)$ is obtained by merging the vertices of each SCC of G (and deleting loops and parallel arcs). For $v \in V$ we denote by v^* the vertex of G^* corresponding to the SCC of G containing v. A SCC is called *sink* if its corresponding vertex in the condensation is a sink.

Similarly to the case of undirected graphs, a closed walk in a digraph G is called an *Eulerian walk* if it uses each arc of G exactly once. A SCC is *Eulerian* if it has an Eulerian walk.

Part I

Seidel's Switching

Chapter 1

Introduction

1.1 Background

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 one by a sequence of switches. The class of graphs that are pairwise switching-equivalent is called a switching class.

The concept of Seidel's switching 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 about two-graphs we refer the reader to the papers of Seidel [Sei76, Sei74, Sei92], Seidel and Taylor [ST81], and Cameron [Cam77]. There are connections of Seidel's switching to other parts of mathematics, such as algebra [Sei76, Cam77, ST81, Wel84, Doo79] and Euclidean geometry [Sei76].

Applications of Seidel's switching can be found in graph algorithms; for example, Seidel's switching plays an important role in Hayward's polynomial-time algorithm for solving the P_3-structure recognition problem [Hay96]. The algorithm of Rotem and Urrutia [RU82] to recognize circular permutation graphs is based on switching, as well as its linear-time variant found later by Srithara-ran [Sri96]. Switching also plays part in the algorithm for bi-join decomposition of graphs by de Montgolfier and Rao [dMR05].

Lindzey [Lin15] noticed that it is possible to speed-up several graph algorithms using switching to a lower number of edges – he obtained up to super-polylogarithmic speed-ups of algorithms for diameter, transitive closure, bipartite maximum matching and general maximum matching. However, he focuses on switching digraphs with a definition somewhat different to Seidel's switching in undirected graphs, and with different properties (a digraph with the minimum number of edges in its switching-class can be found in linear time, which is in contrast with our results contained in this thesis).

Various modifications and generalizations have been studied, such as switching with more than two colours, also of infinite graphs [CT04]; signed switching classes [CWJ86, Wel84]; switching of directed graphs with skew gains [HH00,

EHHR04]. Hage wrote his PhD thesis [Hag01] about switching classes of undirected graphs and about switching classes of directed graphs with skew gains.

In this thesis, we mainly examine Seidel's switching from the computational complexity point of view. In Section 1.2 we summarize the previous results regarding the computational complexity of Seidel's switching. In Section 1.3 we give formal definitions and basic properties of Seidel's switching.

1.2 Computational Complexity of Seidel's Switching

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

Kratochvíl et al. [KNZ92] and also Ehrenfeucht at al. [EHHR00a] compared the computational complexity of the problem if the input graph is switching-equivalent to a graph having a certain property, and the computational complexity of deciding if the input graph itself has the property. They observed that there is no correlation. For example, the problems of deciding if a graph contains a Hamiltonian path or cycle are well known to be NP-complete [GJ79]. However, 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. These results have been extended to graph pancyclicity by Ehrenfeucht et al. [EHHR00b].

On the other hand, the problem of deciding switching-equivalence to a regular graph was proved to be NP-complete by Kratochvíl [Kra03], and switching-equivalence to a k-regular graph for a fixed k is polynomial, while both the regularity and k-regularity of a graph can be tested in polynomial time. Three-colorability and switching-equivalence to a three-colorable graph are both NP-complete [EHHR00a].

Ehrenfeucht et al. [EHHR00a] provided more NP-completeness results: they proved that, given two graphs G and H, it is NP-complete to decide if H is isomorphic to a subgraph of a graph switching-equivalent to G. They further proved a general theorem about the NP-completeness of deciding if a graph is switching-equivalent to a graph containing a large enough induced subgraph with a given property: the property must be *non-trivial* (there exists a graph not having the property, and there exist arbitrarily large graphs having the property), *switching-non-trivial* (there exists a graph that is not switching-equivalent to any graph with the property) and *hereditary* (closed on induced subgraphs). Such properties include being a discrete graph, a complete graph, a bipartite graph, a complete bipartite graph, an acyclic graph, a planar graph, a chordal graph. (However, this does not imply that deciding if the input graph itself is switching-equivalent to a graph with the property is also NP-hard.)

Polynomial-time algorithms are known, in addition to those mentioned above, for the problems of deciding if the input graph is switching-equivalent to a

triangle-free graph [Hay96, HHW02], to an Eulerian graph [HHW02], to a bipartite graph [HHW02], and to a graph with a property that itself can be recognized in polynomial time and that implies minimum degree bounded by a constant [EHHR00a, Kra03].

Somewhat surprisingly, the NP-completeness results have still been rather few. In this thesis, we contribute with more. In Chapter 2 we prove that given a graph G and an integer k, it is NP-complete to decide if G can be switched to contain at most k edges. We further strengthen the result by proving that the problem is NP-complete even for input graphs whose density is bounded by a constant. We thus partially answer a question of Matoušek and Wagner [MW14]. Our results are also related to a problem of Hage [Hag01]. Further, we show a connection of the problem of switching to a low number of edges to coding theory.

In Chapter 3 we focus on the problem if, for a fixed graph H, a given graph is switching-equivalent to an *H-free graph* – a graph that does not contain H as an induced subgraph. Polynomial-time algorithms are known for H isomorphic to K_2 [HHW02], $K_{1,2}$ [KNZ92], K_3 [Hay96, HHW02], P_4 [Her99], $K_{1,3}$ [JK14]. It has been an open problem [KNZ92] whether there exists a graph H such that the problem of switching to an H-free graph is NP-complete. We close this problem by giving infinitely many such graphs H, the smallest of them having nine vertices. We also describe the class of forbidden induced subgraphs for graphs switching-equivalent to a forest of bounded vertex degrees.

1.3 Definitions and Basic Properties

Definition 1.1. Let G be a graph. The *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.,

$$\begin{aligned} V(S(G,v)) &= V(G) \\ E(S(G,v)) &= \big(E(G) \setminus \{vx : x \in V(G)\}\big) \cup \{vx : x \in V(G),\ x \neq v,\ vx \notin E(G)\}. \end{aligned}$$

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 1.2. Let G be a graph. Then the *Seidel's switch of a vertex subset $A \subseteq V(G)$* is denoted by $S(G, A)$, 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.

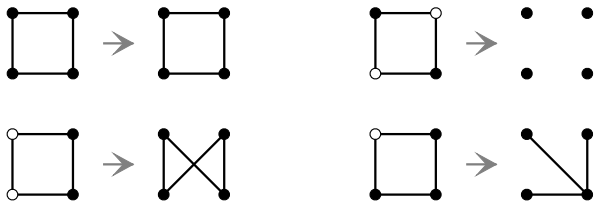


Figure 1.1: Seidel's switches of C_4. Vertices to be switched are white.

To give an example: Fig. 1.1 depicts all the possible switches of C_4 (up to isomorphism). The vertices of the subset A are drawn in white color. 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.

Various basic properties of switching, as stated in the following lemma, can be found in [Hag01].

Lemma 1.3. *Let $G = (V, E)$ be a graph and let $A, B \subseteq V$. Then*

1. $S(G, A)[A] = G[A]$,
2. $S(G, A) = S(G, V \setminus A)$,
3. $S(G, \emptyset) = S(G, V) = G$,
4. $S(S(G, A), B) = S(S(G, B), A) = S(G, A \triangle B)$,
5. $S(S(G, A), A) = G$,
6. $S(\overline{G}, A) = \overline{S(G, A)}$.

Lemma 1.4 (Ehrenfeucht et al. [EHHR00a])**.** *Let $G = (V, E)$ be a graph, let $u \in V$ and $X \subseteq V$. If $u \notin X$, then there exists a unique graph $H \in [G]$ such that $N_H(u) = X$. Namely, it is the graph $S(G, N_G(u) \triangle X)$.*

The following theorem was proved by Ehrenfeucht et al. [EHHR00a] and also independently (in a slightly weaker form) by Kratochvíl [Kra03].

Theorem 1.5. *Let $\mathcal{P}$ be a graph property that can be decided in time $\mathcal{O}(n^a)$ for an integer a. Let every graph on n vertices with the property $\mathcal{P}$ 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 $\mathcal{O}(n^{d(n)+1+\max(a,2)})$.*

We summarize the proofs of Theorem 1.5 by Ehrenfeucht et al. [EHHR00a] and by Kratochvíl [Kra03] – they provide the following simple algorithm which we use later. Let $G = (V, E)$ be the input graph. For all choices of a vertex $v \in V$ and for all choices of a set $X \subseteq (V \setminus v)$ such that $|X| \leq d(n)$ test if the graph $S(G, N_G(v) \triangle X)$ has the property $\mathcal{P}$.

There are $\mathcal{O}(n)$ choices of v, and for each v, there are $\mathcal{O}(n^{d(n)})$ choices of X. To obtain the graph $S(G, N_G(v) \triangle X)$ and to test the property $\mathcal{P}$ takes time $\mathcal{O}(n^{\max(a,2)})$. Hence, the algorithm runs in the given time.

By Lemma 1.4, the graph $S(G, N_G(v) \triangle X)$ is the unique graph in which X is the neighborhood of v; hence, the algorithm tests all the possible switches of G in which there is a vertex of degree at most $d(n)$, and thus, the algorithm gives the correct result.

Chapter 2

Optimizing the Number of Edges

2.1 Switching-Minimality and Related Problems

Hage in his PhD thesis [Hag01, p. 115, Problem 8.5] posed the problem to characterize graphs that have the maximum (or minimum) number of edges in their switching class. We call such graphs *switching-maximal* and *switching-minimal*, respectively.

Properties of switching-maximal graphs were studied by Kozerenko [Koz15]. He proved that any graph with sufficiently large minimum degree is switching-maximal, and that the join of certain graphs is switching-maximal. Further, he gave a characterization of triangle-free switching-maximal graphs and of non-hamiltonian switching-maximal graphs.

It is easy to observe that a graph is switching-maximal if and only if its complement is switching-minimal. We call the problem to decide if a graph is switching-minimal Switch-Minimal.

We mainly focus on the more general problem Switch-Few-Edges – the problem of deciding if a graph can be switched to contain at most a certain number of edges. The paper [JSHK11] (co-authored by the author of this thesis) presented a proof that the problem is NP-complete. Unfortunately, the proof in that paper is not correct. Specifically, Lemma 4.3 of [JSHK11], which claims to establish a reduction from the classical Max-Cut problem to Switch-Few-Edges, is false. The claim of the lemma fails, e. g., on a graph G formed by two disjoint cliques of the same size.

In this chapter (and also in the paper [JJK]), we provide a different proof of the NP-hardness of Switch-Few-Edges, based on a reduction from a restricted version of Max-Cut. Furthermore, we strengthen this result by proving that for any $c > 0$, Switch-Few-Edges is NP-complete even if we require that the input graph has density at most c. We also prove that if the problem Switch-Minimal is co-NP-complete, then for any $c > 0$, the problem is co-NP-complete even on graphs with density at most c.

We thus partially answer a question of Matoušek and Wagner [MW14] posed in connection with properties of simplicial complexes – they asked if deciding switching-minimality was easy for graphs of bounded density. Our results also

indicate that it might be unlikely to get an easy characterization of switching-minimal (or switching-maximal) graphs, which contributes to understanding the question of Hage [Hag01].

In Section 2.2 we present several results about easy special cases of the problems that we focus on. This complements our hardness results – Section 2.3 contains a proof that the problem Switch-Few-Edges is NP-complete (the auxiliary proof of the NP-completeness of Large-Deg-Max-Cut is placed separately in Section 2.4.) In Section 2.5 we show a connection of Switch-Few-Edges to a problem in coding theory. In Section 2.6 we extend the NP-completeness result of Switch-Few-Edges to graphs of bounded density.

2.2 Formal Definitions and Easy Cases

We say that a graph G is $(\leq k)$-*switchable* if there is a set $A \subseteq V(G)$ such that $S(G, A)$ contains at most k edges. Analogously, a graph G is $(\geq k)$-*switchable* if there is a set $A \subseteq V(G)$ such that $S(G, A)$ contains at least k edges. We say that a graph is *switching-reducible* if G is *not* switching-minimal, in other words, if there is a set $A \subseteq V(G)$ such that $S(G, A)$ contains less edges than the graph G.

We define 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?

Switch-Reducible
Input: A graph $G = (V, E)$
Question: Is G switching-reducible?

Switch-Minimal
Input: A graph $G = (V, E)$
Question: Is G switching-minimal?

Switch-Maximal
Input: A graph $G = (V, E)$
Question: Is G switching-maximal?

Lemma 2.1. *The problems* Switch-Few-Edges *and* Switch-Many-Edges *are polynomially equivalent.*

Proof. Let G be a graph. By Lemma 1.3(6) we know 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. □

By an analogous reasoning, we immediately have the first part of the following lemma; the second part is obvious.

Lemma 2.2. *The problems* Switch-Minimal *and* Switch-Maximal *are polynomially equivalent. Moreover,* Switch-Reducible *is the complement of the problem* Switch-Minimal.

Finally, it is easy to see that Switch-Few-Edges is a more general problem than Switch-Reducible. Having an algorithm for Switch-Few-Edges, we can solve Switch-Reducible for a graph G with j edges by running the algorithm on the input $(G, j-1)$ – the graph G is switching-reducible if and only if the algorithm answers "yes".

A simple "brute-force" algorithm for Switch-Few-Edges can be obtained using the proof Theorem 1.5 (on page 18): in a graph with at most k edges all vertex degrees are bounded by k. Hence, we can use $d(n) = k$ and $a = 2$ and get an $\mathcal{O}(n^{k+3})$-time algorithm. In Section 2.3, we provide an NP-completeness proof for Switch-Few-Edges.

The following proposition states a basic relation of switching-minimality and graph degrees.

Proposition 2.3 (folklore)**.** *Every switching-minimal graph $G = (V, E)$ on n vertices has maximum degree at most $\lfloor (n-1)/2 \rfloor$.*

Proof. Clearly, if G contains a vertex v of degree greater than $\lfloor (n-1)/2 \rfloor$, then $S(G, \{v\})$ has fewer edges than G, showing that G is not switching-minimal. □

We remark that for a given graph G we can efficiently construct a switch whose maximum degree is at most $\lfloor (n-1)/2 \rfloor$; one by one, we switch vertices whose degree exceeds this bound (in this way, the number of edges is decreased in each step). However, the graph constructed by this procedure is not necessarily switching-minimal.

For a set $A \subseteq V(G)$, let $e(A)$ denote the number of edges whose one vertex is in A and the other one in $V(G) \setminus A$. The next proposition is an equivalent formulation of Lemma 2.5 of Kozerenko [Koz15], strengthening Proposition 2.3.

Proposition 2.4. *A graph G is switching-minimal if and only if for every $A \subseteq V(G)$, we have*

$$2e(A) \leq |A|(|V(G)| - |A|).$$

We derive the following consequence.

Proposition 2.5. *Let G be a graph with n vertices. If the maximum vertex degree in G is at most $\frac{n}{4}$, then G is switching-minimal.*

Proof. Let A be any subset of $V(G)$. We observe that $e(A) = e(V(G) \setminus A)$; hence we can assume without loss of generality that $|A| \leq n/2$, and thus $|V(G)| - |A| \geq n/2$.

Further, as $e(A) \leq \sum_{v \in A} \deg(v)$, we have that $e(A) \leq |A|\frac{n}{4}$. Hence, $2e(A) \leq |A|(|V(G)| - |A|)$, and the condition of Proposition 2.4 is fulfilled. □

Proposition 2.5 implies that Switch-Few-Edges and Switch-Minimal are trivially solvable in polynomial time for graphs on n vertices with maximum degree at most $\frac{n}{4}$.

We note that in Proposition 2.5, the bound $\frac{n}{4}$ in general cannot be improved, as shown by the example of a k-regular bipartite graph on n vertices with $k > \frac{n}{4}$. Such a graph is switching-equivalent to a $(\frac{n}{2} - k)$-regular bipartite graph, and therefore is not switching-minimal.

2.3 The NP-Completeness of Switch-Few-Edges

We present a proof that the problem Switch-Few-Edges is NP-complete. Our proof corrects the errors contained in the proof of the paper [JSHK11]. The core of the original proof is a reduction from the Max-Cut problem. Our reduction works in a similar way. However, we need the following more special version of Max-Cut (we prove the NP-completeness of Large-Deg-Max-Cut in Section 2.4).

Large-Deg-Max-Cut
Input: A graph G with $2n$ vertices such that the minimum vertex degree of G is $2n-4$ and the complement of G does not contain triangles; an integer j
Question: Does there exist a cut V_1 of $V(G)$ with at least j cut-edges?

Our proof of the NP-completeness of Switch-Few-Edges is based on the following Proposition.

Proposition 2.6. *Let G be a graph. In polynomial time, we can find a graph G' such that $|V(G')| = 4|V(G)|$ and the following statements are equivalent for every integer j:*

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

(b) there exists a set $A \subseteq V(G')$ such that $S(G', A)$ contains at most $|E(G')| - 16j$ edges.

Proof. We first describe the construction of the graph G'. For each vertex u of G we create a corresponding four-tuple $\{u', u'', u''', u''''\}$ of pairwise non-adjacent vertices in G'. An edge of G is then represented by a complete bipartite graph interconnecting the two four-tuples, and a non-edge in G is represented by 8 edges that form a cycle that alternates between the two four-tuples (see Fig. 2.1).

We remark that our construction of G' follows a similar idea as the construction in the attempted proof in the paper [JSHK11], a notable difference being that in the original construction, a vertex of G was represented by a pair of vertices of G' rather than a four-tuple.

A vertex four-tuple in G' corresponding to a vertex of G is called an *o-vertex*. A pair of o-vertices corresponding to an edge of G is called an *o-edge* and a pair of o-vertices corresponding to a non-edge of G is called an *o-non-edge*. An o-vertex

belonging to an o-edge or o-non-edge is called an *end-o-vertex* of that o-edge or o-non-edge. Where there is no danger of confusion, we identify o-vertices with vertices of G, o-edges with edges of G and o-non-edges with non-edges of G. We also speak about *adjacent* and *non-adjacent* o-vertices.

We now prove that the statements (a) and (b) are equivalent. First assume that there is a cut V_1 of $V(G)$ with j' cut-edges. Let V_1' be the set of vertices u', u'', u''', u'''' for all $u \in V_1$. We prove that $S(G', V_1')$ contains $|E(G')| - 16j'$ edges.

We say that a non-edge *crosses the cut* V_1 if the non-edge has exactly one vertex in V_1. It is clear that G' contains 16 edges per every o-edge and 8 edges per every o-non-edge. In $S(G', V_1')$, every o-edge corresponding to an edge that is not a cut-edge is unchanged by the switch and yields 16 edges. Similarly, every o-non-edge corresponding to a non-edge that does not cross the cut yields 8 edges.

Fig. 2.2 illustrates the switches of o-non-edges and o-edges that have exactly one end-o-vertex in V_1. We can see that every o-non-edge corresponding to a non-edge that crosses the cut yields 8 edges in $S(G', V_1')$, and that every o-edge corresponding to a cut-edge yields 0 edges. Altogether, the number of edges of $S(G', V_1')$ differs from $|E(G')|$ exactly by the number of cut-edges multiplied by 16, and hence $S(G', V_1')$ has $|E(G')| - 16j'$ edges, which we wanted to prove.

Now assume that there exists a set $A \subseteq V(G')$ such that $S(G', A)$ contains at most $|E(G')| - 16j$ edges. We want to find a cut in G with at least j cut-edges.

We say that an o-vertex u of G' is *broken in* A if A contains exactly one, two or three vertices out of u', u'', u''', u''''; otherwise, we say that u is *legal in* A. We say that an o-edge or o-non-edge $\{u, v\}$ is *broken in* A if at least one of the o-vertices u, v is broken. Otherwise, we say that $\{u, v\}$ is *legal in* A.

If all vertices of G are legal in A, we say that A is *legal*. Legality is a desired property, because for a legal set A we can define a subset V_A of $V(G)$ such that

$$V_A = \{u \in V(G) : \{u', u'', u''', u''''\} \subseteq A\}.$$

The set V_A then defines a cut in G. If a set is not legal, we proceed more carefully to get a cut from it. For any vertex subset A, we say that a set A' is a *legalization* of A if A' is legal and if A' and A differ only on o-vertices that are broken in A.

We want to show that for every illegal set A, there exists its legalization A' such that the number of edges in $S(G', A')$ is not much higher than in $S(G', A)$. To this end, we give the Algorithm Legalize which for a set A finds a legalization A' of A.

During the run of the Algorithm, we keep an auxiliary set A'' (which we want to become the desired legalization of A). The set A'' is needed in the following

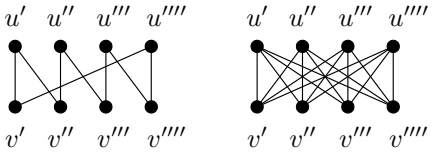


Figure 2.1: The representation of non-edges and edges of G.

Figure 2.2: Switches of an o-non-edge and of an o-edge.

definitions. Let v be an o-vertex and consider the o-vertices that are adjacent to v (through an o-edge); we call them *o-neighbors* of v. The o-neighbors of v are four-tuples of vertices and some of those vertices are in A'', some of them are not. We define $\mathrm{dif}(v)$ as the number of such vertices that are in A'' minus the number of such vertices that are not in A''. (Note that $\mathrm{dif}(v)$ is always an even number, because the total number of vertices in o-neighbors is even. If all o-neighbors were legal, then $\mathrm{dif}(v)$ would be divisible by four.)

The Algorithm is given in Fig. 2.3. In the beginning, the Algorithm sets $A'' := A$, and it is easy to see that in each step the Algorithm changes A'' so that at least one broken o-vertex becomes legal, while o-vertices that are already legal are unchanged. As in the last step the Algorithm legalizes all remaining broken o-vertices, it is clear that the set A'' output by the Algorithm is a legalization of A. We further prove that $|E(S(G', A''))| - |E(S(G', A))| \leq 7$.

We need to introduce more terminology. A pair of vertices of G' which belong to the same o-vertex is called a *v-pair*. A pair of vertices of G' which belong to different o-vertices that are adjacent (in G) is called an *e-pair*. A pair of vertices of G' which belong to different o-vertices that are non-adjacent (in G) is called an *n-pair*. It is easy to see that any edge of G' or $S(G', A'')$ is either a v-pair, an e-pair or an n-pair. We call such edges *v-edges*, *e-edges* and *n-edges*, respectively.

We say that a broken o-vertex v is *asymmetric* if it contains an odd number of vertices of A''; we say that a broken o-vertex is *symmetric* if it contains two vertices of A''.

To measure how the number of edges of $S(G', A'')$ changes during the run of the Algorithm, we define a variable $c(A'')$ which we call the *charge* of the graph $S(G', A'')$. Before the first step we set $c(A'') := |E(S(G', A))|$ (at that moment, A'' is equal to A). After a step of the Algorithm, we update $c(A'')$ in the following way.

- For every v-pair or e-pair that was an edge of $S(G', A'')$ before the step and is no longer an edge of $S(G', A'')$ after the step, we decrease $c(A'')$ by one.
- For every v-pair or e-pair that was not an edge of $S(G', A'')$ before the step and that has become an edge of $S(G', A'')$ after the step, we increase $c(A'')$ by one.
- For every o-vertex v that was legalized in the step and is incident to an o-non-edge, we change $c(A'')$ in the following way:
  - if v is symmetric, we increase $c(A'')$ by 2.5 for every o-non-edge incident to v;

Algorithm Legalize(A)
Set $A'' := A$; do the following while any of the cases applies.

Case 1. There exists a broken o-vertex v such that $|\mathrm{dif}(v)| \geq 4$. If $\mathrm{dif}(v) \geq 4$, set $A'' := A'' \setminus \{v', v'', v''', v''''\}$. Otherwise, set $A'' := A'' \cup \{v', v'', v''', v''''\}$.

Case 2. Case 1 does not apply and there exists an asymmetric broken vertex v such that v contains exactly one vertex from A'' and $\mathrm{dif}(v) = 2$. Set $A'' := A'' \setminus \{v', v'', v''', v''''\}$.

Case 3. Case 1 does not apply and there exists an asymmetric broken vertex v such that v contains exactly three vertices from A'' and $\mathrm{dif}(v) = -2$. Set $A'' := A'' \cup \{v', v'', v''', v''''\}$.

Case 4. None of Cases 1, 2, 3 applies and there exist two adjacent broken o-vertices u and v. Set

$$A_1 := (A'' \cup \{u', u'', u''', u''''\}) \setminus \{v', v'', v''', v''''\},$$

$$A_2 := (A'' \cup \{v', v'', v''', v''''\}) \setminus \{u', u'', u''', u''''\}.$$

If $|E(S(G', A_1))| < |E(S(G', A_2))|$ then set $A'' := A_1$, otherwise set $A'' := A_2$.

Case 5. None of the above cases applies. Then legalize the remaining broken o-vertices arbitrarily, output A'' and STOP.

Figure 2.3: The Algorithm Legalize.

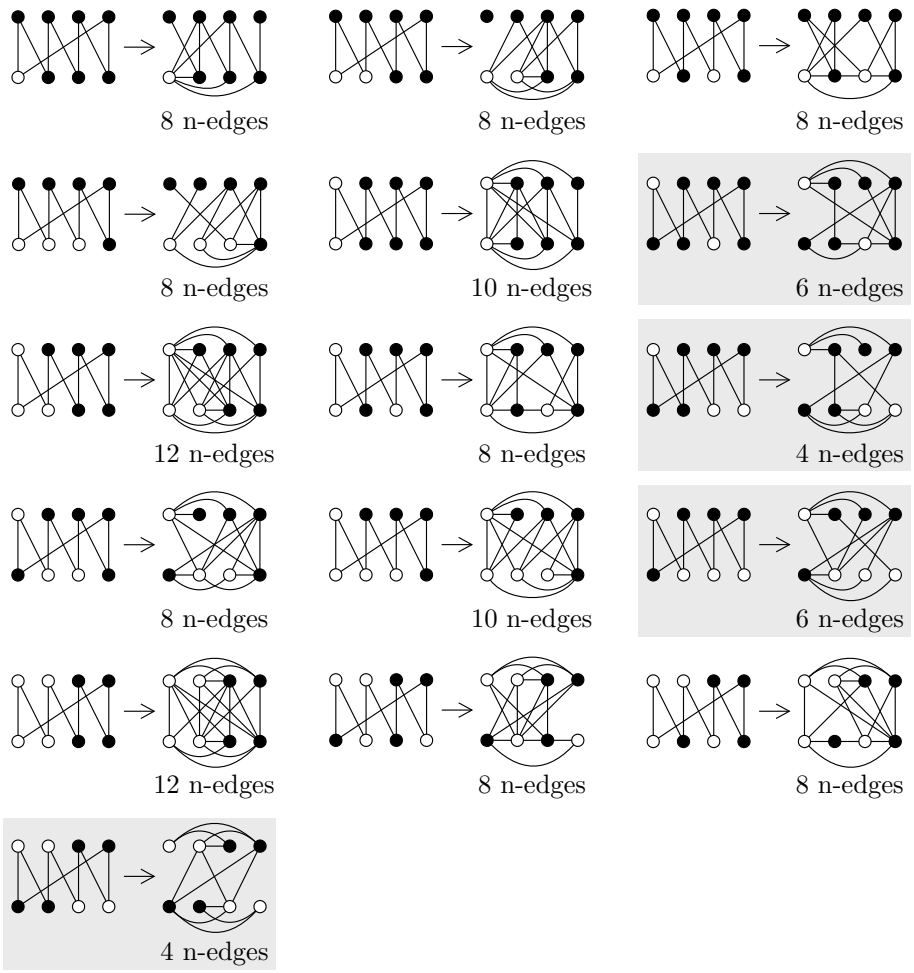


Figure 2.4: All possible illegal switches of o-non-edges (up to symmetry). Vertices of A are marked in white and edges are as in G (left to the arrow) and as in $S(G', A)$ (right to the arrow). In the highlighted cases, the number of n-edges in $S(G', A)$ is lower than 8 for the particular o-non-edge.

– if v is asymmetric, we increase $c(A'')$ by 1.5 for every o-non-edge incident to v.

To explain the last two points, we observe how the number of n-edges increases after legalizing an o-vertex. By analyzing all cases of o-non-edges with one or two broken end-o-vertices (see Fig. 2.4), we get that there are four cases where the o-non-edges have less than 8 n-edges before legalization: these cases are highlighted in Fig. 2.4; there are either 6 or 4 n-edges in each such o-non-edge. In these cases, both end-o-vertices are broken. If there are only 4 n-edges, at least one of the end-o-vertices is symmetric. After one end-o-vertex is legalized, the number of n-edges increases by 2 or 4. When the second end-o-vertex is legalized, the number of n-edges does not increase for this particular o-non-edge.

After both end-o-vertices are legalized, the charge has been changed in the following way: if both end-o-vertices were symmetric, we have increased the charge

by 5. If one of them was symmetric and the other one was asymmetric, we have increased the charge by 4. Finally, if both were asymmetric, we have increased the charge by 3. In all these cases, the increase is an upper bound on the number of contributed n-edges.

Further, every v-edge or e-edge that has appeared or disappeared during the run of the Algorithm is counted immediately after the corresponding step. Hence, we have proved the following Claim.

Claim 2.7. *At the end of the Algorithm we have that* $c(A'') \geq |E(S(G', A''))|$.

Next, we give an upper bound on the charge $c(A'')$.

Claim 2.8. *After every step of the Algorithm except for the last one, the charge* $c(A'')$ *is decreased. After the last step, the charge is increased by at most 7. Hence,* $c(A'') \leq |E(S(G', A))| + 7$.

To prove Claim 2.8, we count how the charge changes after each step of the Algorithm Legalize. We distinguish cases according to which the step was done.

Case 1. We may assume without loss of generality that $\mathrm{dif}(v) \geq 4$ (otherwise we swap the roles of A'' and $V(G') \setminus A''$. Further, v can be either symmetric or asymmetric; we first assume that v is symmetric (see Fig. 2.5). Then by its legalization the number of v-edges is decreased by 4.

As $\mathrm{dif}(v) \geq 4$, then among vertices in o-neighbors of v, there must be at least four more vertices belonging to A'' than those not belonging to A''. Thus, by removing any vertex of $\{v', v'', v''', v''''\}$ from A'' we reduce the number of e-edges by at least 4. As v contains two vertices out of $\{v', v'', v''', v''''\}$, we reduce the number of e-edges by at least 8.

For n-pairs that have one vertex inside v the charge is increased by at most $3 \cdot 2.5$, which is 7.5. To sum it up:

- For v-pairs the charge is decreased by 4,
- for e-pairs the charge is decreased by at least 8,
- for n-pairs the the charge is increased by at most 7.5.

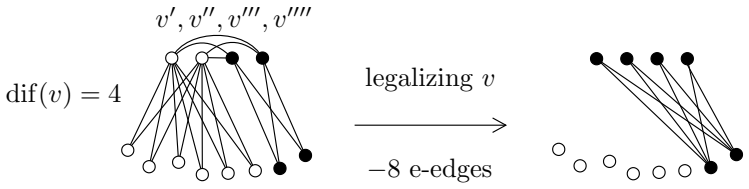


Figure 2.5: A simplified illustration to the analysis of Case 1. Vertices of A'' are marked in white, and edges are as in $S(G', A'')$ before the step (left to the arrow) and after the step (right to the arrow).

Altogether, the charge is decreased by at least 4.5.

If the o-vertex v is asymmetric, then in an analogical way we have that for v-pairs the charge is decreased by 3, for e-pairs the charge is decreased by at least 4, and for n-pairs the the charge is increased by at most 4.5. Altogether, the charge is decreased by at least 2.5.

Case 2. The analysis is similar as above. We get that the charge is decreased by 3 for v-pairs, decreased by 2 for e-pairs, and increased by at most 4.5 for n-pairs. Altogether, the charge is decreased by at least 0.5.

Case 3. This case is symmetric to Case 2. Hence, the charge is decreased by at least 0.5 as well.

Case 4. In this case, when counting how the charge was changed because of e-pairs, we need to bound both the number of e-edges between a vertex in u and a vertex in v, and the number of e-edges between a vertex inside u or v and a vertex inside one of their other o-neighbors. This depends also on the values of $\mathrm{dif}(u)$ and $\mathrm{dif}(v)$.

We analyze four subcases of o-edges whose both end-o-vertices are broken – they are numbered as in Fig. 2.7.

I. First assume that $\mathrm{dif}(u) = 0$ and $\mathrm{dif}(v) = 0$ (see Fig. 2.6). We can see that vertices inside v contribute by -2 to $\mathrm{dif}(u)$. Hence, outside v, there must be two more vertices in o-neighbors of u that are in A'' than those not in A''. The same holds symmetrically for o-neighbors of v outside u.

We may without loss of generality assume that the Algorithm chose to set

$$A'' := (A'' \cup \{v', v'', v''', v''''\}) \setminus \{u', u'', u''', u''''\}.$$

Then, the number of e-pairs adjacent to both u and v is decreased by 10; the number of e-pairs adjacent to u and not to v is decreased by 2, and the number of e-pairs adjacent to v and not to u is increased by 6. Altogether, the charge is decreased by 6 for e-pairs.

For v-pairs, the charge is decreased by 6, and for n-pairs, the charge is increased by at most $6 \cdot 1.5$. Altogether, the charge is decreased by at least 3.

It remains to analyze the cases when $\mathrm{dif}(u)$ and $\mathrm{dif}(v)$ are different. As neither Case 2 nor Case 3 applies, we know that none of $\mathrm{dif}(u)$, $\mathrm{dif}(v)$ is equal to 2.

By analogical ideas as above, we get that if one of $\mathrm{dif}(u)$, $\mathrm{dif}(v)$ is equal to 0 and the other one to -2, the charge is decreased by at least 9. If both $\mathrm{dif}(u)$, $\mathrm{dif}(v)$ are equal to -2, then the charge is decreased by at least 7.

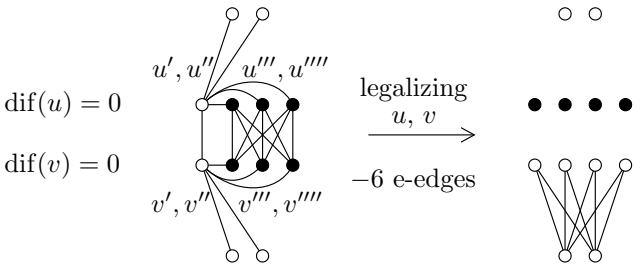


Figure 2.6: A simplified illustration to the analysis of Case 4, I. Vertices of A'' are marked in white, and edges are as in $S(G', A'')$ before the step (left to the arrow) and after the step (right to the arrow).

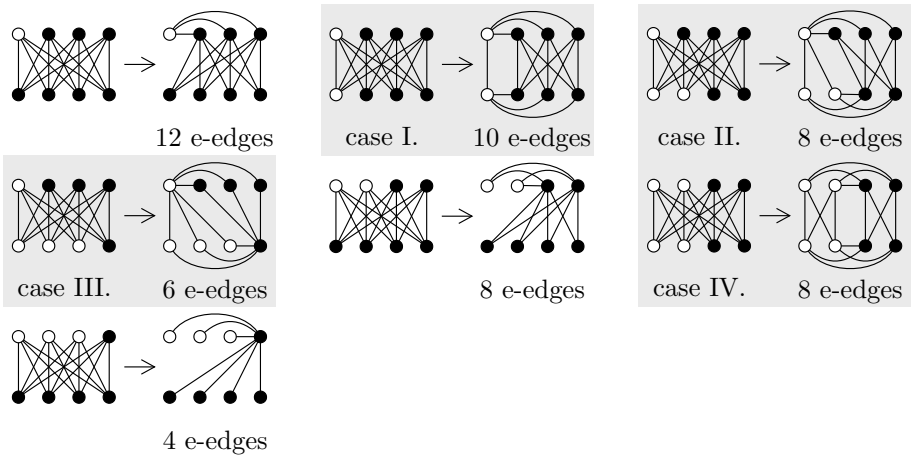


Figure 2.7: All possible illegal switches of o-edges (up to symmetry). Vertices of A are marked in white and edges are as in G (left to the arrow) and as in $S(G', A)$ (right to the arrow). In the highlighted cases, both end-o-vertices are broken.

II. As u is asymmetric and v is symmetric, we have that for n-pairs the charge is increased by at most $3 \cdot 2.5 + 3 \cdot 1.5$, which is 12. For v-pairs, the charge is decreased by $3 + 4$, which is 7.

We consider the case when the Algorithm chose to set

$$A'' := A'' \cup \{u', u'', u''', u''''\} \setminus \{v', v'', v''', v''''\}$$

(if we get a sufficient bound for this case, then the other case could only be better).

To count the decrease for e-pairs, we need to consider the values of $\mathrm{dif}(u)$ and $\mathrm{dif}(v)$. Assume that $\mathrm{dif}(u) = 0$. Then, outside v, there must be the same number of vertices in o-neighbors of u that are in A'' as those that are not in A''.

If $\mathrm{dif}(v) = 0$, then outside u there must be two more vertices in o-neighbors of v that are in A'' than those not in A''. Then for e-pairs, the charge is decreased by 12. If $\mathrm{dif}(v) = 2$, then using analogous ideas we get that for e-pairs, the charge is decreased by 16. If $\mathrm{dif}(v) = -2$, we get 8.

Now assume that $\mathrm{dif}(u) = -2$. By considering the number of vertices in o-neighbors of u and v, we get that the charge decrease for e-pairs is either 14 (if $\mathrm{dif}(v) = -2$) or 18 (if $\mathrm{dif}(v) = 0$) or 22 (if $\mathrm{dif}(v) = 2$).

As Case 2 does not apply, we know that $\mathrm{dif}(u)$ is not equal to 2. Hence, we have considered all the cases, and the charge decrease for e-pairs is at least 8. Altogether, the charge is decreased by at least $-12 + 7 + 8$, which is 3.

III. As both u and v are asymmetric, we have that for n-pairs the charge is increased by at most $6 \cdot 1.5$, which is 9. For v-pairs, the charge is decreased by $3 + 3$, which is 6.

Again, we consider the case when the Algorithm chose to set

$$A'' := A'' \cup \{u', u'', u''', u''''\} \setminus \{v', v'', v''', v''''\}.$$

By using the same idea as above, we get that for e-pairs, the charge is decreased by 18 (if $\mathrm{dif}(u) = 0$ and $\mathrm{dif}(v) = 0$), or by 24 (if $\mathrm{dif}(u) = -2$ and $\mathrm{dif}(v) = 0$, or if $\mathrm{dif}(u) = 0$ and $\mathrm{dif}(v) = 2$), or by 30 (if $\mathrm{dif}(u) = -2$ and $\mathrm{dif}(v) = 2$).

As Case 2 does not apply, we know that $\mathrm{dif}(u)$ cannot be 2 and $\mathrm{dif}(v)$ cannot be -2. Hence, we have considered all the cases, and for e-pairs the charge is decreased by at least 18. Altogether, the charge is decreased by at least $-9 + 6 + 18$, which is 15.

IV. As both u and v are symmetric, we have that for n-pairs the charge is increased by at most $6 \cdot 2.5$, which is 15. For v-pairs, the charge is decreased by $4 + 4$, which is 8.

Without loss of generality, we consider only cases when $\mathrm{dif}(u) \leq \mathrm{dif}(v)$ (the other cases are symmetric). Thus, we may limit ourselves again to the case when the Algorithm chose to set

$$A'' := A'' \cup \{u', u'', u''', u''''\} \setminus \{v', v'', v''', v''''\}.$$

If $\mathrm{dif}(u) = \mathrm{dif}(v)$, then we easily check that the charge decrease for e-pairs is 8. If $\mathrm{dif}(u) = 0$ and $\mathrm{dif}(v) = 2$ then the charge decrease for e-pairs is 12. If $\mathrm{dif}(u) = -2$ and $\mathrm{dif}(v) = 0$ then the decrease is 12, and if $\mathrm{dif}(u) = -2$ and $\mathrm{dif}(v) = 2$ then the decrease is 16.

Altogether, the charge decrease for e-pairs is at least 8, and the total decrease is at least $-15 + 8 + 8$, which is 1.

Case 5. If Case 5 applies, then all remaining broken o-vertices must be pairwise non-adjacent (because Case 4 does not apply). Hence, there must be at most two broken o-vertices left (otherwise, there would be a triangle in the complement of the input graph, which would contradict the assumptions). Further, each of these o-vertices has dif $= 0$, because all its o-neighbors are legal and Case 1 does not apply. Thus, the charge change for e-pairs due to this last step is 0.

To count the charge change for n-pairs and v-pairs, we analyze the five cases (one or two o-vertices, symmetric or asymmetric). If there is one symmetric o-vertex left, then the charge increase for n-pairs is at most $3 \cdot 2.5$ and the decrease for v-pairs is 4, hence the total increase is at most $7.5 - 4$, which is 3.5. If there is one asymmetric o-vertex, then the total increase is at most $3 \cdot 1.5 - 3$, which is 1.5.

If there are two broken o-vertices left and both are asymmetric, then the total increase is at most $6 \cdot 1.5 - 6$, which is 3. If one of them is symmetric and the other one is asymmetric we get 5; if both are symmetric, we get 7. Altogether, we get that the charge is increased by at most 7.

We have proved Claim 2.8. Further, by Claim 2.7 and Claim 2.8 we have that $|E(S(G', A''))| \leq |E(S(G', A))| + 7$, and hence A'' is the sought legalization of A.

We continue the proof of Proposition 2.6. We have already argued that a legal set A'' defines a subset $V_{A''}$ of $V(G)$, and hence a cut in G. Assume that $\mathrm{cutset}(V_{A''})$ has j' edges. From the proof of the first implication of Proposition 2.6 we know that the number of edges in $S(G', A'')$ can be expressed as $|E(G')| - 16j'$.

On the other hand, we have proved that the number of edges in $S(G', A'')$ is at most $|E(G')| - 16j + 7$. We get that $|E(G')| - 16j' \leq |E(G')| - 16j + 7$, and hence $j' \geq j - 7/16$. As both j and j' are integers, we have that $j' \geq j$. Hence, $\mathrm{cutset}(V_{A''})$ has at least j edges, and Proposition 2.6 is proved. □

Theorem 2.9. Switch-Few-Edges *is NP-complete.*

Proof. Theorem 2.11 in the next section gives the NP-completeness of Large-Deg-Max-Cut. Further, by Proposition 2.6, an instance (G, j) of Large-Deg-Max-Cut can be transformed into an instance (G', j') of Switch-Few-Edges such that there is a cut in G with at least j cut-edges if and only if G' is $(\leq j')$-switchable. The transformation works in polynomial time.

Finally, it is clear that the problem Switch-Few-Edges is in NP. □

2.4 The NP-Completeness of Large-Deg-Max-Cut

In this section, we prove the NP-completeness of the auxiliary problem Large-Deg-Max-Cut which is used in the proofs of Section 2.3. We repeat the statement of the problem here.

Large-Deg-Max-Cut
Input: A graph G with $2n$ vertices such that the minimum vertex degree of G is $2n-4$ and the complement of G is connected and does not contain triangles; an integer j
Question: Does there exist a cut V_1 of G with at least j cut-edges?

The proof of the NP-completeness of Large-Deg-Max-Cut uses a reduction from the Connected-Min-Bisection problem stated below. We first define some needed notions.

Let G be a graph with $2n$ vertices. A *bisection of* G is a partition S_1, S_2 of $V(G)$ such that $|S_1| = |S_2| = n$ (hence, a bisection is a special case of a cut). The size of cutset(S_1) is called the *size* of the bisection S_1, S_2. A *minimum bisection of* G is a bisection of G with minimum size. We define the following problems.

Min-Bisection
Input: A graph G with $2n$ vertices, an integer b
Question: Is there a bisection S_1, S_2 of $V(G)$ such that cutset(S_1) contains at most b edges?

Connected-Min-Bisection
Input: A connected graph G with $2n$ vertices, an integer b
Question: Is there a bisection S_1, S_2 of $V(G)$ such that cutset(S_1) contains at most b edges?

Garey et al. [GJS76] proved that Min-Bisection is NP-complete (by a reduction of Max-Cut). We remark that their version of the minimum bisection problem (called "Minimum Cut Into Equal-Sized Subsets") is slightly different from Min-Bisection as stated here – in the version of Garey et al. [GJS76], two distinguished vertices must be each in one part of the partition. However, it is immediate that the two distinguished vertices are not important in the proof, and that their proof gives the NP-completeness of Min-Bisection as well.

Furthermore, their reduction from Max-Cut (see [GJS76, pages 242–243])

produces only connected graphs as instances of the minimum bisection problem. Hence, their proof also yields the NP-completeness of Connected-Min-Bisection.

From the NP-completeness of Min-Bisection, Bui et al. [BCLS87] proved the NP-completeness of Min-Bisection restricted to 3-regular graphs (as a part of a more general result, see [BCLS87, proof of Theorem 2]). We use the construction from their proof to prove the NP-completeness of Large-Deg-Max-Cut. The following Lemma provides a relation of the size of the minimum bisection in a special graph and the size of the maximum cut in the complement of the graph.

Lemma 2.10. *Let G be a connected 3-regular graph on $2n$ vertices. Let b be the size of the minimum bisection in G and let c be the size of the maximum cut in $\overline{G}$. Then $b = n^2 - c$.*

Proof. Let S_1, S_2 be a minimum bisection in G and let b be the size of the bisection. In $\overline{G}$, the partition S_1, S_2 yields a cut with $n^2 - b$ cut-edges. Hence, $c \geq n^2 - b$.

On the other hand, let V_1, V_2 be a maximum cut in $\overline{G}$ for which the sizes of V_1 and V_2 are as close as possible. If $|V_1| = |V_2| = n$, the partition $|V_1|$, $|V_2|$ gives a bisection in G of size $n^2 - c$, hence $b \leq n^2 - c$ and we are done.

Otherwise, assume that $|V_1| = n - k$ and $|V_2| = n + k$ for a $k \geq 1$. As the graph G is connected, there is a vertex v in V_2 that has at least one neighbor in V_1. We set $V_1' = V_1 \cup \{v\}$ and $V_2' = V_2 \setminus \{v\}$.

By the choice of v, there is at least one edge between v and V_1 in G, and thus in $\overline{G}$, there are at most $n - k - 1$ edges between v and V_1; hence, there are at most $n - k - 1$ cut-edges adjacent to v in $\overline{G}$.

As $\deg_G(v) = 3$, we have that v has at most two neighbors in V_2 and at least $n + k - 3$ non-neighbors in V_2. Hence, in the partition V_1', V_2', there will be at least $n + k - 3$ cut-edges adjacent to v in $\overline{G}$. Cut edges that are not adjacent to v are the same in V_1, V_2 as in V_1', V_2'.

Altogether, $|\text{cutset}(V_1')| - |\text{cutset}(V_1)| \geq n + k - 3 - (n - k - 1) \geq 2k - 2 \geq 0$. Hence, the partition V_1', V_2' has smaller difference of the sizes of the two parts while the size of the cut is not smaller, which is a contradiction with the choice of V_1, V_2. □

Theorem 2.11. Large-Deg-Max-Cut *is NP-complete.*

Proof. Let (G, b) be an instance of Connected-Min-Bisection. We use the construction of Bui et al. [BCLS87, proof of Theorem 2]. Their first step is to construct from an instance (G, b) of Min-Bisection a 3-regular graph G^* such that G has a minimum bisection of size b if and only if G^* has a minimum bisection of size b. Further, it is immediate from their construction that G^* contains no triangles, and if G is connected, then G^* is connected as well. Moreover, G^* has an even number of vertices.

We see that $\overline{G^*}$ fulfills the conditions of an instance of Large-Deg-Max-Cut. By Lemma 2.10 we know that G^* has a minimum bisection of size b if and only if $\overline{G^*}$ has a maximum cut of size $m^2 - b$.

Altogether, G has a minimum bisection of size b if and only if $\overline{G^*}$ has a maximum cut of size $m^2 - b$. Hence, $(\overline{G^*}, m^2 - b)$ is an equivalent instance of Large-Deg-Max-Cut. To finish the proof that Large-Deg-Max-Cut is NP-complete, we observe that Large-Deg-Max-Cut is in NP. □

2.5 A Connection to Coding Theory

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 aim to show the connection to coding theory without giving many details or proofs.

A *characteristic vector* of a subset B of a set A is the vector $\mathbf{x}_B = (x_a)_{a \in A}$ such that $x_a = 1$ if $a \in B$ and $x_a = 0$ if $a \notin B$. We also denote the a-th coordinate of $\mathbf{x}_B$ by $\mathbf{x}_B(a)$. Let $\mathbb{Z}_2$ be the finite field with two elements, 0 and 1. A characteristic vector may be seen as a vector over $\mathbb{Z}_2$. In this section, all vectors are considered to be vectors over $\mathbb{Z}_2$, and the summation of vectors is done over $\mathbb{Z}_2$.

The *Hamming weight* of a vector $\mathbf{v}$ is the number of 1-s in $\mathbf{v}$, and the *Hamming distance* of two vectors $\mathbf{v}$ and $\mathbf{w}$ is the number of coordinates where the vectors differ.

Let V be a fixed set of n vertices, and let E be a set of edges on V, i. e., $E \subseteq \binom{V}{2}$. The set E can be represented by the characteristic vector $\mathbf{x}_E$ of the set E. Thus, the graph $G = (V, E)$ is represented by a vector of length $\binom{n}{2}$. The following observation expresses how Seidel's switching works by means of characteristic vectors.

Observation 2.12. *Let V be a set of vertices, let $K_n = (V, \binom{V}{2})$ be the complete graph on the set V, let V_1, V_2 be a partition of V, and let $S = \mathrm{cutset}(V_1)$ in K_n. Then for any $G = (V, E)$,*

$$\mathbf{x}_{E(S(G,V_1))} = \mathbf{x}_{E(S(G,V_2))} = \mathbf{x}_E + \mathbf{x}_S.$$

Therefore, if we seek a switch of G with the minimum number of edges, we seek a vertex subset V_1 such that the characteristic vector $\mathbf{x}_E + \mathbf{x}_{\mathrm{cutset}(V_1)}$ has the minimum Hamming weight. Or, equivalently, we seek a cut-set S in K_n with the minimum Hamming distance between $\mathbf{x}_S$ and $\mathbf{x}_E$.

It is a well-known fact (see, for example, [GY05]) that the set of all characteristic vectors of cut-sets in a graph G is a vector space; it is called the *cut space* and denoted by $\mathcal{C}^*(G)$. The cut space is orthogonal to the *cycle space* $\mathcal{C}(G)$, which is the set of characteristic vectors of Eulerian subgraphs if G. It is also known that the dimension of $\mathcal{C}^*(G)$ is $|V| - c$, where c is the number of connected components of G, and that the dimension of $\mathcal{C}(G)$ is $|E(G)| - |V(G)| + c$.

A *linear code* of length n and rank k is a vector subspace C of $\mathcal{F}^n$, where $\mathcal{F}$ is a finite field, and such that the dimension of C is k. Such a code is also called a linear $[n, k]$ code. Elements of C are also called *codewords*. A linear code can

be characterized by a *parity-check matrix* H: the codewords are exactly vectors $\mathbf{v}$ such that $H\mathbf{v} = \mathbf{0}$.

For a connected graph G, the cut space $\mathcal{C}^*(G)$ can also be viewed as a linear $[|E|, |V| - 1]$ code with a parity-check matrix H 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 independently by Kasami [Kas61], Huffman [Huf64], Frazer [Fra64], and Hakimi and Frank [HF65].

Linear codes are used as error correcting codes – suppose that we have transmitted a codeword over a channel and some coordinates might have changed due to communication noise; each coordinate has changed with the same low probability. We want to find a codeword that is closest to the received vector, because this is most likely to be the original codeword. This is an important problem in coding theory.

Formally, assume that the code is given by a binary $p \times q$ matrix H and let w be an integer. We are given a vector $\mathbf{v} \in \mathbb{Z}_2^q$ and we seek a codeword $\mathbf{c} \in \mathbb{Z}_2^q$ such that the Hamming distance of $\mathbf{c}$ and $\mathbf{v}$ is at most w. We may reformulate the problem using the following observation (the vector $\mathbf{e}$ represents the "erroneous coordinates" by which $\mathbf{v}$ differs from a codeword).

Observation 2.13. *The following conditions are equivalent.*

1. *There exists a vector $\mathbf{c} \in \mathbb{Z}_2^q$ such that $H\mathbf{c} = \mathbf{0}$ and the Hamming distance of $\mathbf{c}$ and $\mathbf{v}$ is at most w,*
2. *there exists a vector $\mathbf{e} \in \mathbb{Z}_2^q$ such that the Hamming weight of $\mathbf{e}$ is at most w, and $H\mathbf{e} = H\mathbf{v}$.*

Proof. We set $\mathbf{e} = \mathbf{c} + \mathbf{v}$ to prove the first implication, and $\mathbf{c} = \mathbf{e} + \mathbf{v}$ to prove the other one. □

The problem was proved to be NP-complete by Berlekamp et al. [BMvT78] as the following decision problem (in view of Observation 2.13 we can see why this is "maximum likelihood decoding").

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

We prove that SWITCH-FEW-EDGES is a special case of MAXIMUM-LIKELIHOOD-DECODING.

Proposition 2.14. 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.*

Proof. Having a graph $G = (V, E)$, we want to find a set $A \subseteq V(G)$ such that the graph $S(G, A)$ contains at most k edges. Let K_n be the complete graph on

the set V; we set H to be the parity-check matrix of $\mathcal{C}^*(K_n)$. Further, we set $\mathbf{r} = H\mathbf{x}_E$, and $w = k$.

Let $\mathbf{e} \in \mathbb{Z}_2^q$ be a vector of Hamming weight at most w such that $H\mathbf{e} = \mathbf{r}$. Then $H\mathbf{e} = H\mathbf{x}_E$, and hence $H(\mathbf{e} + \mathbf{x}_E) = \mathbf{0}$. Let $\mathbf{a} = \mathbf{e} + x_E$. As $H\mathbf{a} = \mathbf{0}$, the vector $\mathbf{a}$ is an element of the cut space $\mathcal{C}^*(K_n)$, and there exists a set $A \subseteq V(G)$ such that $\mathbf{a}$ is the characteristic vector of cutset(A).

By Observation 2.12 we have that $\mathbf{x}_E + \mathbf{a}$ is the characteristic vector of $S(G, A)$. Further, as the Hamming weight of $\mathbf{e}$ is at most k, we have that the Hamming weight of $\mathbf{x}_E + \mathbf{a}$ is at most k. Thus, the graph $S(G, A)$ has at most k edges, and we are done. □

Special cases of Maximum-Likelihood-Decoding have been studied. It is known that the problem is NP-complete even if we allow an unbounded time for the preprocessing of 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 that Switch-Few-Edges is NP-complete provides an alternative proof of Bruck and Naor's result by using K_n as the fixed graph.

2.6 Switching of Graphs with Bounded Density

The *density* of a graph G is defined as

$$D(G) = \frac{|E(G)|}{\binom{|V(G)|}{2}} = \frac{2|E(G)|}{|V(G)|(|V(G)| - 1)}.$$

In connection with properties of simplicial complexes, Matoušek and Wagner [MW14] asked if deciding switching-minimality was easy for graphs of bounded density. We give a partial negative answer by proving that the problem Switch-Few-Edges stays NP-complete even for graphs of density bounded by an arbitrarily small constant. This is in contrast with Proposition 2.5, which shows that any graph G with maximum degree at most $|V(G)|/4$ is switching-minimal. The core of our argument is the following Proposition.

Proposition 2.15. *Let G be a graph, let k be an integer, and let c be a fixed constant in $(0, 1)$. In polynomial time, we can find a graph G' and an integer k' such that*

1. *$D(G') \leq c$,*
2. *G' is $(\leq k')$-switchable if and only if G is $(\leq k)$-switchable,*
3. *G' is switching-minimal if and only if G is switching-minimal, and*
4. *$|V(G')| = O(|V(G)|)$.*

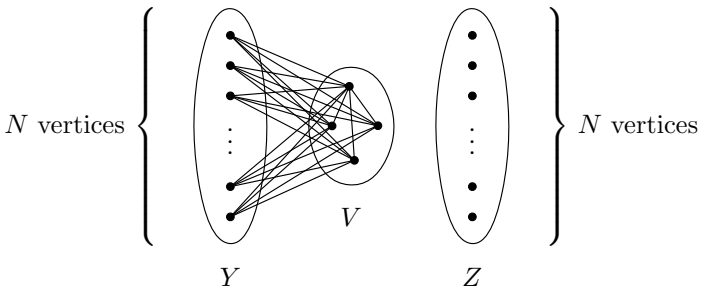


Figure 2.8: The graph G'.

Proof. Let $n = |V(G)|$ and let $N = \max\left\{n, \left\lceil \frac{3n}{4c} \right\rceil\right\}$. We construct the graph G' in the following way (see also Fig. 2.8). Let $V = V(G)$. Then

$$V(G') = V \cup Y \cup Z,$$

where Y is a set of N new vertices and Z is a set of N more new vertices, and

$$E(G') = \{\{v_1, v_2\} \colon v_1 \in Y, v_2 \in V\} \cup E(G).$$

We prove that G' fulfills the conditions of Proposition 2.15. It is easy to see that Condition 4 holds and that G' can be obtained in polynomial time. We prove that Conditions 2 and 3 hold, too.

Assume that G is switching-reducible, i. e., there exists a set $A \subseteq V$ such that $S(G, A)$ contains fewer edges than G. Let us count the number of edges in $S(G', A)$.

It is easy to see that if we switch a subset of V in G', the number of edges whose one endpoint is outside V is unchanged, and the number of edges with both endpoints outside V remains zero. We also observe that $S(G', A)[V]$ (the induced subgraph of $S(G', A)$ on the vertex subset V) is equal to $S(G, A)$. Hence, $S(G', A)$ has fewer edges than G', showing that G' is switching-reducible.

Moreover, if $S(G, A)$ has l edges for an integer l, then $S(G', A)$ has $l + nN$ edges. Thus, if G is $(\leq k)$-switchable, we have that G' is $(\leq k + nN)$-switchable.

Now assume that G' is switching-reducible, i. e., there exists a set $A \subseteq V(G')$ such that $S(G', A)$ has fewer edges than G'. If $A \subseteq V$, we have that $S(G, A)$ has fewer edges than G, and Condition 3 is satisfied. On the other hand, if $A \not\subseteq V$, we use the following Claim.

Claim 2.16. *Let A be a subset of $V(G')$ and let $A' = A \cap V$. Then the number of edges in $S(G', A')$ is less than or equal to the number of edges in $S(G', A)$.*

To prove the claim, we fix a set $A \subseteq V(G')$. We may assume that

$$|A \cap (Y \cup Z)| \leq |Y \cup Z|/2 = N,$$

otherwise we replace A by its complement $\overline{A} = V(G') \setminus A$ (note that $S(G', A \cap V)$ has the same number of edges as $S(G', \overline{A} \cap V)$).

We define the sets $A' = A \cap V$ and $A'' = A \setminus A' = A \cap (Y \cup Z)$. Let $G'_1 = S(G', A')$ and $G'_2 = S(G', A)$. Note that $G'_2 = S(G'_1, A'')$. To prove the claim, we need to show that G'_1 has at most as many edges as G'_2.

In G'_2, every vertex of A'' is adjacent to every vertex of $(Y \cup Z) \setminus A''$, whereas no such pair is adjacent in G'_1. This means that

$$|E(G'_2) \setminus E(G'_1)| \geq |A''|(|Y| + |Z| - |A''|) \geq |A''|N,$$

where we used the fact that A'' has size at most N.

On the other hand, an edge belonging to G'_1 but not to G'_2 must necessarily connect a vertex from A'' with a vertex from V. Therefore, $|E(G'_1) \setminus E(G'_2)| \leq |A''|n$. Combining these estimates, we get

$$\begin{aligned} |E(G'_2)| - |E(G'_1)| &= |E(G'_2) \setminus E(G'_1)| - |E(G'_1) \setminus E(G'_2)| \\ &\geq |A''|N - |A''|n \\ &\geq 0. \end{aligned}$$

This proves the claim. As a consequence of Claim 2.16, if G' is switching-reducible, then it can be reduced by switching a set $A' \subseteq V$. The same set A' then reduces G, and Condition 3 of the Proposition holds. Analogically, if G' can be switched to contain L edges for an integer L, then G can be switched to contain $L - nN$ edges. Hence, we have proved Condition 2 with $k' = k + nN$.

It remains to check Condition 1. By definition, the density of G' is

$$\begin{aligned} D(G') &= \frac{2|E(G')|}{(2N+n)(2N+n-1)} \\ &\leq \frac{2\left(\binom{n}{2} + nN\right)}{(2N+n)(2N+n-1)} \\ &\leq \frac{n^2 + 2nN}{4N^2} \\ &\leq \frac{3nN}{4N^2} = \frac{3n}{4N} \\ &\leq c. \end{aligned}$$

This completes the proof. □

Proposition 2.15 allows us to state a stronger version of Theorem 2.9 for the special case of graphs with bounded density.

Theorem 2.17. *For every $c > 0$, the problem* Switch-Few-Edges *is NP-complete for graphs of density at most c.*

Proof. As shown by Proposition 2.15, a general instance (G, k) of Switch-Few-Edges can be transformed into an equivalent instance (G', k') of density at most c. Since Switch-Few-Edges is NP-complete on general instances by Theorem 2.9, it remains NP-complete on instances of density at most c. □

2.7 Concluding Remarks

To conclude, we mention some related open problems.

Problem 2.18. We have been trying to prove that the problem Switch-Reducible is NP-complete (and hence, Switch-Minimal is co-NP-complete). We have not yet succeeded. However, if it is true, then Proposition 2.15 gives the following analogue of Theorem 2.17 even for these problems.

Proposition 2.19. *If the problem* Switch-Reducible *is NP-complete, then for every $c > 0$, the problem* Switch-Reducible *is NP-complete for graphs of density at most c, and the problem* Switch-Minimal *is co-NP-complete for graphs of density at most c.*

Problem 2.20. Let $d > 0$ be a constant. What can we say about the complexity of Switch-Reducible and Switch-Few-Edges on graphs of maximum degree at most dn? If $d \leq \frac{1}{4}$, the two problems are trivial by Proposition 2.5. On the other hand, for $d \geq \frac{1}{2}$ the restriction on maximum degree becomes irrelevant, since any switching-minimal graph has maximum degree at most $\frac{n}{2}$ by Proposition 2.3. For any $d \in (\frac{1}{4}, \frac{1}{2})$, the complexity of the two problems on instances of maximum degree at most dn is open.

Chapter 3

On Switching to H-Free Graphs

3.1 Previous Results

Let H be a graph. We say that a graph G is *H-free* if G does not contain H as an induced subgraph. In this chapter, we focus on the problem whether a graph is switching-equivalent to an H-free graph.

Polynomial-time algorithms for this problem for several fixed graphs H have already been known. The algorithm for K_2 is simple (see [HHW02]), the one for $K_{1,2}$ is due to Kratochvíl et al. [KNZ92]. Hayward [Hay96] and independently Hage at al. [HHW02] found an algorithm for K_3; the result is a core of the polynomial-time algorithm for recognizing P_3-structures of graphs. The case of P_4 has been solved by Hertz [Her99] in connection to the so called perfect switching classes. An algorithm for $K_{1,3}$ was part of my master thesis [Ond06].

From basic properties of Seidel's switching (see Lemma 1.3) it can be observed that an algorithm for H, when run on the complement of the input graph, gives an algorithm for $\overline{H}$. (The switching-equivalence of H and H', however, does not yield any obvious relation of algorithms for H and H'.)

As listed above, only polynomial-time variants of switching to H-free graphs have been identified. It has been an open problem [KNZ92] whether there exists a graph H such that the problem of switching to an H-free graph is NP-complete. In Section 3.4 we solve this problem by giving infinitely many such graphs H, the smallest of them having nine vertices.

We say that P is a *graph property* if P is an isomorphism-closed class of graphs; a class of graphs is *hereditary* if it is closed on induced subgraphs. As we explain in Section 3.2, for a hereditary graph property there exists a class $\mathcal{F}(S(P))$ of minimal forbidden induced subgraphs such that a graph G is switching-equivalent to a graph with P if and only if G does not contain any element of $\mathcal{F}(S(P))$ as an induced subgraph. If $\mathcal{F}(S(P))$ is finite, this can be tested in polynomial time.

Hence, when examining the computational complexity of switching-equivalence to a graph with P, the characterization by $\mathcal{F}(S(P))$ is valuable. However, a polynomial-time algorithm may exist even when $\mathcal{F}(S(P))$ is infinite. And, in some cases, more efficient algorithms are already known.

Hage and Harju [HH04] give the class $\mathcal{F}(S(\text{“being acyclic”}))$. Apart from

graphs switching-equivalent to cycles C_n for $n \geq 7$, there are 905 graphs in the class, each having at most nine vertices (they are partitioned into 24 switching classes). A computer program was employed to obtain this result.

Characterizations by a finite class $\mathcal{F}$ are known for switching-equivalence to H-free graphs. Namely, for H isomorphic to K_2 the class $\mathcal{F}(S(\text{“being } H\text{-free”}))$ consists of two graphs on three vertices [HHW02]. For P_4 it consists of four graphs on five vertices, as shown by Hertz [Her99].

As we have already mentioned, there exists a polynomial-time algorithm to decide if a graph is switching-equivalent to a K_3-free graph [Hay96, HHW02] or to a $K_{1,3}$-free graph [Ond06, JK14]. However, the classes of forbidden induced subgraphs are not known to be finite.

There are also slightly different results – characterizations by the class $\mathcal{F}$ for switching classes whose *all* graphs have a certain hereditary property, namely, for switching classes whose all graphs are perfect (due to Hertz [Her99]), and for switching classes whose all graphs contain a 1-perfect code (due to Kratochvíl [Kra89]).

In my master thesis [Ond06], I have described the class $\mathcal{F}(S(\text{“being } H\text{-free”}))$ for $H = K_{1,2}$ by ten forbidden induced subgraphs, each having five vertices (the graphs are partitioned into two switching classes). This characterization does not bring any algorithmic improvement for deciding switching-equivalence to 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. [KNZ92] runs in time $\mathcal{O}(n^3)$. However, we find this characterization interesting, because the minimal forbidden induced subgraphs are few and small; for $H = K_3$, there are at least hundreds of graphs in $\mathcal{F}$ [Ond06].

More generally, we may study switching-equivalence to $\mathcal{H}$-free graphs, i. e., graphs that are H-free for every $H \in \mathcal{H}$, where $\mathcal{H}$ is a fixed graph class. If we consider the class $\mathcal{C}$ of all cycles, then $\mathcal{C}$-freeness is the same as acyclicity. For switching-equivalence to acyclic graphs, the set $\mathcal{F}$ was described by Hage and Harju [HH04] in a result mentioned above.

We remark that any subclass of acyclic graphs is 1-degenerated; assume that it can be decided in time $\mathcal{O}(n^a)$ whether a graph belongs to the subclass. Then it follows from Theorem 1.5 that the problem if a graph is switching-equivalent to a graph from the subclass can be decided in time $\mathcal{O}(n^{2+max(a,2)})$.

When $\mathcal{H} = \mathcal{C} \cup \{K_{1,d+1}\}$, the $\mathcal{H}$-free graphs are forests of maximum degree at most d; in case that $d = 1$, they are called partial matchings. For switching-equivalence to partial matchings, the set $\mathcal{F}$ was described by Herman [Her06]. In Section 3.3 we extend this result to any integer $d \geq 5$, and give a superset of $\mathcal{F}$ for $d = 2, 3, 4$.

In Section 3.2, we formally define the classes $\mathcal{F}(P)$ and $\mathcal{F}(S(P))$ for a hereditary graph property P. In Section 3.3, as we mention above, we describe the class $\mathcal{F}(S(\text{“being acyclic and } K_{1,d+1}\text{-free”}))$ for $d \geq 5$, and we give a superset of the class for $d = 2, 3, 4$. In Section 3.4 we describe infinitely many graphs H such that the problem of switching to an H-free graph is NP-complete.

3.2 Hereditary Properties and Forbidden Induced Subgraphs

For a hereditary graph property P we define the class $\mathcal{F}(P)$ of minimal forbidden induced subgraphs for P in the following way:

$$\mathcal{F}(P) = \{F : F \notin P, \forall F' < F : F' \in P\}.$$

Then a graph G has the property P if and only if it does not contain any element of $\mathcal{F}$.

Lemma 3.1. *Let P be a hereditary graph property. Then the property of being switching-equivalent to a graph with P is hereditary as well.*

Proof. Let G be a graph, let $A \subseteq V(G)$ such that $S(G, A)$ has P. Clearly, for any $G' \leq G$, it holds that $S(G', A \cap V(G')) \leq S(G, A)$. Hence, by hereditariness, $S(G', A \cap V(G'))$ has P, so G' is also switching-equivalent to a graph with P. □

In view of Lemma 3.1 we may further define the class $\mathcal{F}(S(P))$ as the class of minimal forbidden induced subgraphs for switching-equivalence to P.

We study the property of being H-free for fixed graphs H. It can easily be observed that this property is hereditary; our aim is to find the class $\mathcal{F}(S(\text{“being } H\text{-free”}))$. Similarly as for algorithms, there is a relation between results for H and for $\overline{H}$. From Lemma 1.3 we get the following corollary.

Proposition 3.2. *For every graph F it holds that $F \in \mathcal{F}(S(\text{“being } H\text{-free”}))$ if and only if $\overline{F} \in \mathcal{F}(S(\text{“being } \overline{H}\text{-free”}))$.*

Proof. Assume that a graph F is an element of $\mathcal{F}(S(\text{“being } H\text{-free”}))$, but $\overline{F}$ is not in $\mathcal{F}(S(\text{“being } \overline{H}\text{-free”}))$. Then $\overline{F}$ is either not forbidden or not minimal.

In the former case, $\overline{F}$ is switching-equivalent to an $\overline{H}$-free graph. But then, by Proposition 1.3, F is switching-equivalent to an H-free graph, which is a contradiction. In the latter case, we get in the same way a contradiction with the minimality of F. The other implication follows from the fact that $\overline{\overline{F}} = F$. □

3.3 Switching to a d-Forest

In this section we describe the class $\mathcal{F}(S(\text{“being acyclic and } K_{1,d+1}\text{-free”}))$. An acyclic $K_{1,d+1}$-free graph is a forest with all degrees at most d; we call it a *d-forest*. As we have already mentioned, for $d = 1$ the class was described by Herman [Her06]. Hence, in this section we consider the case of $d \geq 2$. Let d be fixed.

For a graph G, we say that a set $A \subseteq V(G)$ is *feasible* if $S(G, A)$ is a d-forest. Fig. 3.1 contains examples of graphs with feasible sets. These graphs will be important in the further proofs.

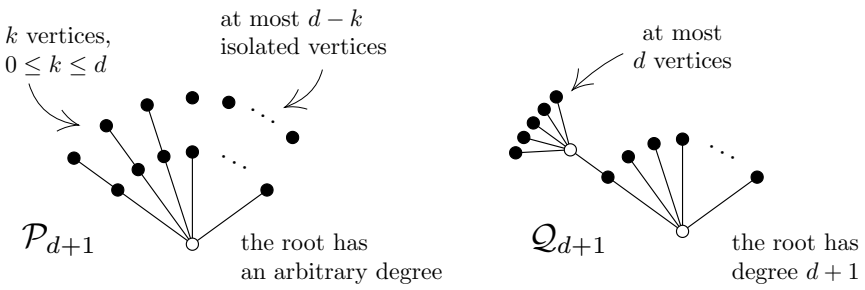


Figure 3.1: Graph classes $\mathcal{P}_{d+1}$ and $\mathcal{Q}_{d+1}$. Vertices of feasible sets are marked in white.

A graph isomorphic to $K_{1,d+1}$ is called a *multiclaw*. We say that vertices of degree one in a multiclaw are *nails*, and the vertex of degree $d+1$ is the *spur*. A vertex non-adjacent to any vertex of a multiclaw M is called a *crumb* of M.

The following lemma shows a simple useful property of feasible sets.

Lemma 3.3. *Let G contain an induced multiclaw M, and let A be a feasible set. Then A satisfies the following conditions.*

1. *A contains either*
   - *all nails and not the spur, or equivalently*
   - *the spur and no nails.*
2. *Furthermore, any crumb of M is an element of A if and only if the nails of M are.*

Proof. Since $S(G,A)$ is a d-forest, $S(M,A)$ is not a multiclaw. Thus, $A \cap V(M)$ is a proper nonempty subset of $V(M)$.

Assume that A violates condition (1). Then, using the simple fact that $S(G,A) = S(G, V(G) \setminus A)$, we may assume that A contains the spur s, a nail x, and it does not contain a nail y. There are at least three nails in a multiclaw, because $d \ge 2$. If A contains no more nails, then $S(M,A)$ is again a multiclaw (whose spur is now x). Otherwise, A contains another nail z; then the vertices s, x, y, z induce a cycle in $S(M,A)$, which is a contradiction.

It remains to prove condition (2). Suppose the contrary: without loss of generality, A contains no nails and it does contain a crumb c. Then the vertex c together with the nails forms another multiclaw in $S(G,A)$, which is not possible. □

The graphs $A^0_{d+1}, \ldots, A^{d+1}_{d+1}$, B_{d+1}, C_{d+1}, D_{d+1}, E_{d+1} in Fig. 3.2 are called the *fan graphs*.

Lemma 3.4. *The fan graphs $A^0_{d+1}, \ldots, A^{d+1}_{d+1}$, B_{d+1}, C_{d+1}, D_{d+1}, E_{d+1} are minimal forbidden induced subgraphs for switching to d-forests.*

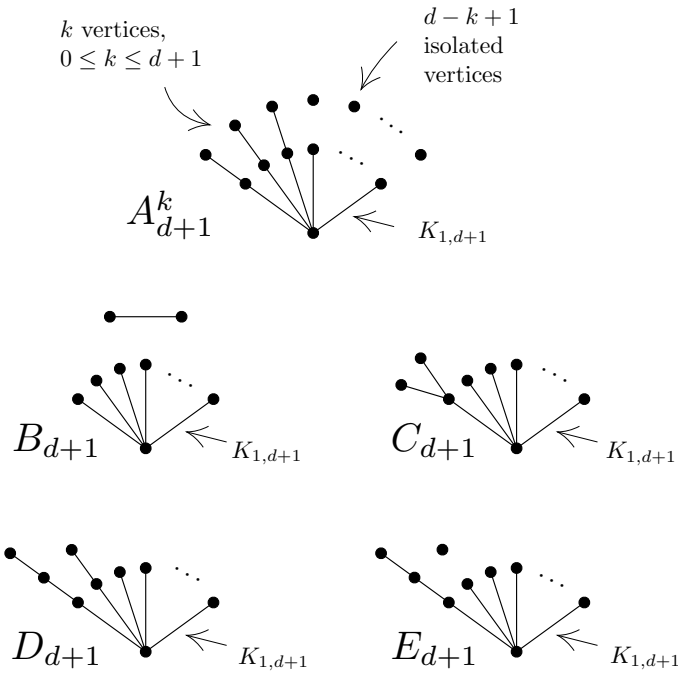


Figure 3.2: The fan graphs.

Proof. To prove the minimality, it suffices to show that by removing any vertex from any fan graph we obtain a graph that is switching-equivalent to a d-forest.

All fan graphs are acyclic. It is straightforward to verify that removing a vertex from any fan graph either destroys all occurrences of induced $K_{1,d+1}$ in it (and thus makes it a d-forest), or makes it an element of $\mathcal{P}_{d+1}$ or $\mathcal{Q}_{d+1}$ (these classes are defined in Fig. 3.1). The elements of $\mathcal{P}_{d+1}$ and $\mathcal{Q}_{d+1}$ can be switched to a d-forest in the way indicated in Fig. 3.1.

To prove that the graphs are forbidden, we show that none of them can be switched to a d-forest. For contradiction, assume that a fan graph has a feasible set A. We consider only feasible sets that satisfy the conditions of Lemma 3.3.

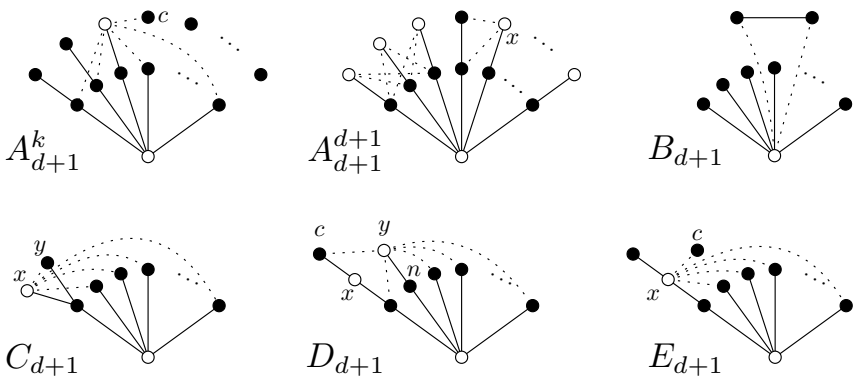


Figure 3.3: Illustration of the proof of Lemma 3.4. Vertices of A are marked white, the others black.

Moreover, we assume that A contains the bottom spur of the considered graph, and it does not contain any of its nails nor any crumbs. A vertex not contained in a multiclaw is called *loose*.

The following cases are illustrated in Fig. 3.3.

- Any graph A^k_{d+1}, where $0 \leq k < d+1$, contains at least one crumb c. Then if any loose non-crumb is in A, it yields a multiclaw together with its non-adjacent nails and with c. Hence A contains the spur only, but that also yields a multiclaw in $S(A^k_{d+1}, A)$ (with the same spur).

- The graph A^{d+1}_{d+1} contains no crumbs; any loose vertex is adjacent to a nail. If A contains the spur only, then there is a multiclaw in $S(A^{d+1}_{d+1}, A)$. Hence A contains at least one loose vertex x. But then all the remaining loose vertices are in A, because the converse would yield a triangle (as indicated in Fig. 3.3). Since $d \geq 2$, there are at least three loose vertices in A, and those vertices together with their adjacent nails yield a cycle of length six.

- In B_{d+1}, the above assumptions determine the set A entirely. However, $S(B_{d+1}, A)$ then contains a triangle, and is not a d-forest.

- C_{d+1} contains two loose vertices, x and y. If none of them is in A, then they form a cycle in $S(C_{d+1}, A)$ with the spur and their adjacent nail n. If both are in A, then they yield a four-cycle with any two nails non-adjacent to them. Hence one of them, say, x, is in A, and y is not. But then x forms a multiclaw in $S(C_{d+1}, A)$ together with y and all the formerly non-adjacent nails.

- In D_{d+1}, the neighbor x of the crumb c must be in A, otherwise x, c and the spur form a triangle in $S(D_{d+1}, A)$ (see Fig. 3.3). Then y is also in A, otherwise there is a triangle x, y, n. But in that case there is a multiclaw on y, c and all the nails except for n.

- In E_{d+1}, the vertex x is in A for the same reasons as in D_{d+1}. Then x, the crumb c and the nails formerly non-adjacent to x yield a multiclaw in $S(E_{d+1}, A)$.

We conclude that, indeed, no fan graph can be switched to a d-forest, and that the fan graphs are minimal forbidden subgraphs for switching to a d-forest. □

The following result was proved by Hage and Harju [HH04].

Theorem 3.5. *The class $\mathcal{F}(S(\text{“being acyclic”}))$ consists of 27 switching classes of graphs having at most 9 vertices, and of switching classes $[C_n]$ of cycles for $n \geq 10$.*

Let $\mathcal{A}$ denote the class $\mathcal{F}(S(\text{“being acyclic”}))$, and let $\mathcal{F}_{d+1}$ denote the union of switching classes of the fan graphs $A^0_{d+1}, \ldots, A^{d+1}_{d+1}$, B_{d+1}, C_{d+1}, D_{d+1}, E_{d+1}.

Theorem 3.6. *For $d \geq 5$, the class $\mathcal{F}(S(\text{"being a } d\text{-forest"}))$ equals $\mathcal{A} \cup \mathcal{F}_{d+1}$. For $d \in \{2, 3, 4\}$, the class $\mathcal{F}(S(\text{"being a } d\text{-forest"}))$ is a subset of $\mathcal{A} \cup \mathcal{F}_{d+1}$.*

Proof. Let $d \geq 2$. For simplicity, we denote the class $\mathcal{F}(S(\text{"being a } d\text{-forest"}))$ by $\mathcal{F}$. We first prove that $\mathcal{F} \subseteq \mathcal{A} \cup \mathcal{F}_{d+1}$.

Let G be a graph that contains no element of $\mathcal{A}$ nor $\mathcal{F}_{d+1}$ as an induced subgraph. We prove that G can be switched to a d-forest. As G contains no element of $\mathcal{A}$ as an induced subgraph, then it is switching-equivalent to an acyclic graph; we denote the acyclic graph by G^a.

If G^a contains no $K_{1,d+1}$, then we are done. Otherwise, we fix a copy of $K_{1,d+1}$ in G^a; let x be the spur of the copy. The connected component of G^a containing x is a tree; we root the tree in x and we organize the vertices of the connected component in *levels* according to their distance from x. (Thus, the neighbors of x are the first level, etc.) We distinguish the following two cases.

- There is a vertex v in the third level; let w be its ancestor. If the degree of the root is at least $d + 2$, then the vertex v with w, the root and the first-level vertices non-adjacent to w form a B_{d+1}, which is a contradiction. Hence, the degree of the root is exactly $d + 1$.

  Apart from w, there is no other second-level vertex, otherwise G^a contains a C_{d+1} or a D_{d+1}. Furthermore, G^a is connected, otherwise G^a contains an E_{d+1}. There are at most d vertices in the third level, because they are all non-adjacent to the first level, and there may be no A^0_{d+1} in G^a. Altogether, we have that G^a is an element of $\mathcal{Q}_{d+1}$.

- There is no vertex in the third level. Since G^a does not contain an A^0_{d+1}, there are at most $d + 1$ connected components in it. And since G^a does not contain a B_{d+1}, all components except for the one containing x are just isolated vertices (crumbs). Moreover, any branching in the first level would imply a C_{d+1}. Thus, each first-level vertex has at most one descendant. And finally, there is no A^0_{d+1}, so the number of second-level vertices plus the number of crumbs is at most d. Therefore, G^a is an element of $\mathcal{P}_{d+1}$.

In both cases, G^a can be switched to a d-forest in the way depicted in Fig. 3.1. Thus, the graph G is switching-equivalent to a d-forest as well.

To prove the statement of the theorem, it remains to prove that if $d \geq 5$, then $\mathcal{A} \cup \mathcal{F}_{d+1} \subseteq \mathcal{F}$. By Lemma 3.4, $\mathcal{F}_{d+1} \subseteq \mathcal{F}$. We prove that $\mathcal{A} \subseteq \mathcal{F}$, too.

Let F be a graph from the class $\mathcal{A}$. We know that F cannot be switched to an acyclic graph, hence it is also forbidden for switching to a d-forest; we prove that it is minimal.

Let $\mathcal{C} = \bigcup_{n \geq 10}[C_n]$. It is easy to see that no proper induced subgraph of a graph in $\mathcal{C}$ is forbidden; hence the graphs in the class $\mathcal{C}$ are indeed minimal. Now let $F \in \mathcal{A} \setminus \mathcal{C}$. Assume that F is not minimal; let F' be a proper induced subgraph of F such that F' is minimal forbidden for switching to a d-forest. F' is not an element of $\mathcal{A}$, and it does not contain any element of $\mathcal{A}$ as an induced subgraph, because $\mathcal{A}$ contains no proper induced subgraph of F.

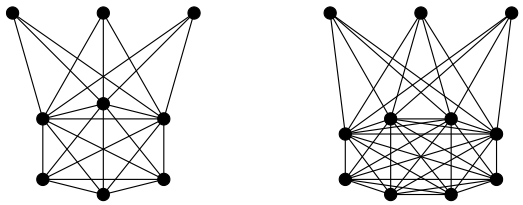


Figure 3.4: Hedgehogs H_3 and H_4.

By Theorem 3.5 the graph F has at most 9 vertices, hence F' has at most 8 vertices. As $d \geq 5$, the graphs of $\mathcal{F}_{d+1}$ have at least 9 vertices. Hence, F' does not contain any element of $\mathcal{F}_{d+1}$ as an induced subgraph.

As F' contains no element of $\mathcal{A} \cup \mathcal{F}_{d+1}$ as an induced subgraph, then by the arguments above, F' can be switched to a d-forest, which is a contradiction with the choice of F'. We have proved that if $d \geq 5$ then $\mathcal{A} \cup \mathcal{F}_{d+1} \subseteq \mathcal{F}$, which finishes the proof. □

3.4 Switching to a Hedgehog-Free Graph

In this section, we provide infinitely many graphs H such that deciding if a graph can be switched to an H-free graph is NP-complete. For each $k \geq 3$ we define a graph called *hedgehog* on $2k + 3$ vertices, denoted by H_k. It is composed of a clique on $2k$ vertices called the *body* of the hedgehog, and three other vertices called *pins*. Exactly k vertices of the body are adjacent to all pins, and the remaining k vertices of the body are non-adjacent to any pin (see Fig. 3.4).

Our proof that switching to an H_k-free graph is NP-complete is a reduction from the following problem which is known to be NP-complete [GJ79].

Monotone-Not-All-Equal-3-SAT
Input: A formula in CNF, in which every clause contains exactly three variables and there are no negations.
Question: Does there exist a truth assignment such that in each clause, at least one variable is true and at least one is false?

Clearly, we may assume that the input formulas contain at least eight distinct variables.

Theorem 3.7. *For each $k \geq 3$ and each hedgehog H_k, it is NP-complete to decide if a given graph can be switched to an H_k-free graph.*

Proof. Consider a fixed $k \geq 3$, and let φ be an instance of Monotone-Not-All-Equal-3-SAT with at least eight distinct variables. We transform φ into a graph $G = G_\varphi$ as follows. For each variable v, there is a vertex x_v. A clause $c = (v_i \vee v_j \vee v_l)$ is represented by two disjoint cliques K_c and K'_c, each having $4k-2$ vertices. Exactly $2k-1$ vertices of K_c and $2k-1$ vertices of K'_c are adjacent to x_{v_i}, x_{v_j} and x_{v_l}, and are called *outer vertices*. Other vertices of K_c and K'_c are called *inner*. See Fig. 3.5 for illustration.

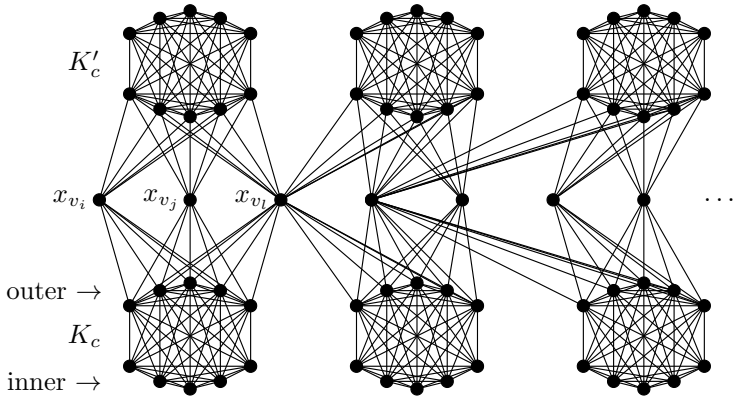


Figure 3.5: An example of the graph G_φ (for $k = 3$). The cliques K_c and K'_c represent a clause $c = (v_i \vee v_j \vee v_l)$.

Claim 3.8. *If φ is satisfiable, then G is switching-equivalent to a H_k-free graph.*

Let μ be a satisfying valuation of φ, and let A be the subset of vertices corresponding to true variables, i. e., $A = \{x_v : \mu(v) = \text{true}\}$. We prove that $S(G, A)$ is H_k-free.

We use the following terminology: vertices in A are called *switched*, other vertices of G are called *non-switched*. Vertices of G that correspond to a variable of φ are called *variable vertices*, others are called *non-variable vertices*.

Assume for contradiction that there is a hedgehog H_k in $S(G, A)$.

As none of the non-variable vertices is switched, it is clear that they induce the same cliques in $S(G, A)$ as in G, and that vertices in distinct cliques are not adjacent. Hence, the hedgehog body contains vertices of at most one such clique. At the same time, it contains vertices of at least one such clique, because variable vertices form a complete bipartite graph in $S(G, A)$, so at most two variable vertices can be in the body. We denote this clique by K, the set of inner vertices of K by I, and the set of outer vertices of K by O.

Let c be the clause represented by K. We further denote the set of non-switched variable vertices representing the variables (literals) of c as L, the set of other non-switched variable vertices as N (non-literals). L_A and N_A denote the corresponding sets of switched variable vertices.

Since variable vertices induce a complete bipartite subgraph in $S(G, A)$, the hedgehog body contains at most two variable vertices, namely, one in N_A and one in L. We denote them by x_{N_A} and x_L, respectively. The remaining body vertices are elements of $O \cup I$.

Now consider pins of the hedgehog; they are adjacent to exactly k body vertices out of the total $2k$. The adjacency of vertices of $S(G, A)$ to the body is as follows.

- Vertices in L are adjacent to vertices in O and to x_{N_A},
- vertices in L_A are adjacent to vertices in I and to x_L,

- vertices in N are adjacent to x_{N_A},
- vertices in N_A are adjacent to vertices in O, I, and to x_L,
- vertices of other cliques are adjacent only to x_{N_A} and/or x_L.

We observe that in the latter three cases, vertices have either too few or too many neighbors in the body, as $k > 2$. A vertex of the first case can be a pin, so can a vertex of the second case, but not at the same time, as their neighborhoods in the body are disjoint.

Then, the pins are either three vertices of L or three vertices of L_A. In both cases, the variables of the clause c are all assigned the same value, and c is unsatisfied, which is a contradiction. Thus, we have proved Claim 3.8.

Claim 3.9. *If G is switching-equivalent to a H_k-free graph, then φ is satisfiable.*

Let A be a vertex subset of G such that $S(G, A)$ is H_k-free. We define a valuation μ of variables so that $\mu(v)$ is true if and only if $x_v \in A$ (there may also be non-variable vertices in A, but these do not influence μ).

Assume that a clause $c = (v_i \vee v_j \vee v_l)$ is unsatisfied in μ. We prove that there is a hedgehog H_k in $S(G, A)$ whose body is in the union of vertices of K_c and K'_c, and whose pins are variable vertices.

K_c and K'_c are two cliques in G and regardless of A, their vertices induce two cliques in $S(G, A)$ as well. One of these cliques contains at least $4k - 2$ vertices. We denote this clique by K.

Consider the role of a vertex of K in G and its relation to the set A. It is either an outer vertex of a clique and not an element of A, or an inner vertex and not in A, or outer in A, or inner in A. We denote the sets of these vertices of K by O, I, O_A, and I_A, respectively.

Similarly, we divide all variable vertices according to whether they are literals or non-literals of the clause c, and whether they belong to A, into sets L, N, L_A, and N_A.

In $S(G, A)$, the adjacency between vertices of K and variable vertices is as follows ("a" meaning adjacent and "n" meaning non-adjacent).

	L	N	N_A	L_A
O	a	n	a	n
I	n	n	a	a
I_A	a	a	n	n
O_A	n	a	n	a

To find a hedgehog, we discuss the sizes of the sets above. As we assume that the clause c is unsatisfied, the three vertices x_{v_i}, x_{v_j} and x_{v_l} are either all in L, or all in L_A. We consider the first case, the second being symmetrical. Then the size of L is three.

Vertices in O and I_A are adjacent to vertices in L, vertices in I and O_A are not. Hence, if there are at least k vertices in $O \cup I_A$, and at least k vertices in $I \cup O_A$, then these $k + k$ vertices form a hedgehog with pins in L.

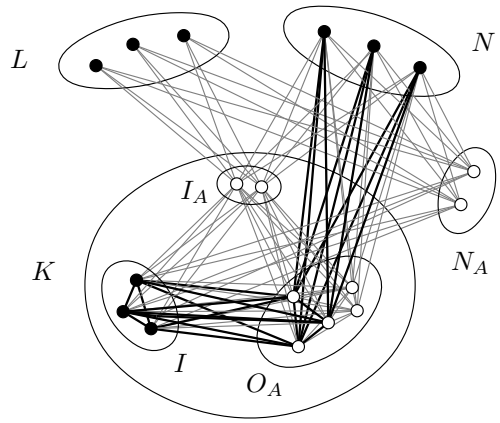


Figure 3.6: A hedgehog in $I \cup O_A \cup N$ in $S(G, A)$. The sets O and L_A are empty. Vertices in A are white.

Otherwise, assume that $|O \cup I_A|$ is at most $k-1$. Then $|I \cup O_A|$ is at least $3k-1$. By the construction, each of I and O_A contains at most $2k-1$ vertices. Then the sizes of O_A and I are at least k, and these vertices form the body of a hedgehog. The pins are either in N or in N_A; we have assumed that there are at most eight distinct variables in φ, and at most three of them occur in the clause c, hence either N or N_A contains at least three vertices. Fig. 3.6 shows an example of a hedgehog with pins in N.

If, on the other hand, $|I \cup O_A|$ is at most $k-1$, we proceed symmetrically with the role of outer and inner vertices interchanged. Again, we find a hedgehog with pins in N or in N_A.

Hence, as $S(G, A)$ is H_k-free, no clause can be unsatisfied. Then μ is a satisfying valuation, and φ is satisfiable, which finishes the proof of Claim 3.9.

Having proved Claim 3.8 and Claim 3.9, we are almost done with the proof of Theorem 3.7. Clearly, the transformation of φ to G can be done in polynomial time. It is easy to see that the problem is in NP. A certificate containing the set A can be checked in polynomial time – we simply test each induced subgraph of $S(G, A)$ having $2k+3$ vertices, whether it is a hedgehog. □

We remark that unless P=NP, Theorem 3.7 implies that for each $k \geq 3$ the class $\mathcal{F}(S(\text{“being } H_k\text{-free”}))$ is infinite – if the class was finite, then there would be a “brute-force” polynomial-time algorithm to decide if a graph is switching-equivalent to an H_k-free graph.

3.5 Concluding Remarks

We have examined the problem of switching-equivalence to an H-free graph for fixed graphs H. Polynomial-time decision algorithms are known for this problem if H has at most three vertices or is isomorphic to P_4, to $K_{1,3}$, or to their complements. In Section 3.4 we have given infinitely many graphs H for which

the problem is NP-complete. However, we are still very far from a complete characterization of graphs H.

Problem 3.10. For which graphs H is the problem to decide if a graph is switching-equivalent to an H-free graph solvable in polynomial time?

Part II

Matchings Under Preferences

Chapter 4

Introduction

4.1 Matchings Under Preferences

A *matching* in a graph is a subset of its edges such that no two of them share any vertex. The problem of finding a matching of maximum size is a well-studied problem. There are polynomial-time algorithms known for finding a maximum matching in general graphs [Edm65, MV80] or in the special case of bipartite graphs, such as the Hopcroft–Karp algorithm [HK73] or various maximum network flow algorithms [FF56, FF10, Din70, EK72, GR97, GT88] which solve a generalization of the problem in an efficient way.

The task to find a matching occurs in many practical applications: assignment of students to universities, families to housing, kidney transplant patients to donors, children to schools, junior doctors to hospitals, etc. However, in a number of such natural settings, participants have preferences over the outcomes and the goal is to find an assignment that optimizes the satisfaction of the participants. Then it might happen that a classical matching algorithm cannot be used.

In 1962, Gale and Shapley [GS62] introduced the Stable Marriage problem – the problem to match men to women under certain conditions of stability. Since then, the Stable Marriage problem and similar matching problems have been subject to a lot of research from the point of view of various areas, such as game theory and economics, computational social choice fields, algorithms and complexity. Gale and Shapley [GS62] designed the well-known Gale-Shapley algorithm for the Stable Marriage problem. This algorithm has been put to practical use in a wide range of large-scale applications in many countries.

In 1989, Gusfield and Irving published a monograph "The Stable Marriage Problem: Structure and Algorithms" [GI89]. This book covered the structural and algorithmic knowledge on stable matching problems of that time. Another monograph – "Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis" by Roth and Sotomayor [RS90] – was published in 1990. This book focuses on the game-theoretic point of view.

Roth and Shapley, who are both leading figures in the research area of matchings under preferences, were awarded the Nobel Prize in Economic Sciences in 2012, "for the theory of stable allocations and the practice of market de-

sign" [NOB].

For a comprehensive overview of the current state of the art concerning algorithmic aspects of matchings under preferences, see the recent book "Algorithmics of Matching under Preferences" by Manlove [Man13].

Matching problems with preferences can be classified into three classes: bipartite matching problems with two-sided preferences; bipartite matching problems with one-sided preferences; non-bipartite matching problems with preferences. In this thesis, we deal with problems belonging to the second class. These can be seen as problems of allocating a set of indivisible goods to agents, where each agents ranks the goods in order of preference. Example applications include campus housing allocation [CS02, PPR08], DVD rental markets [ACKM06], assigning conference papers to reviewers [GKK+10].

We focus on the setting where each agent initially owns one good which is called a "house" - this problem is known as the Housing Market problem [SS74, RP77, Rot82]. (The variant where there are no initial property rights is called the House Allocation problem [HZ79, Zho90, ACMM05]).

Various notions of optimality have been considered, such as Pareto optimality [ACMM05, CF10], core matchings [SS74, RP77, Wak84, QW04], economic equilibrium [SS74, FSW03, DPS03].

The housing market model can be used in applications with possibly very large input instances – for example, Yuan [Yua96] describes large residence exchange fairs for subsidized public housing in China (80 000 person-attendances in Beijing, May 1991). There are centralized matching schemes that work by collecting all input data and then computing an optimal matching by running an algorithm; this algorithm must be able to process all data efficiently. Hence, the computational complexity point of view is important.

In this thesis, we address the computational complexity of finding an economic equilibrium in a housing market. Namely, we deal with a variant of the problem proposed by Fekete et al. [FSW03] where houses are not necessarily unique: some houses have the same *type* and must then have equal price, and agents are indifferent between houses of the same type.

In Section 4.2 we describe the Housing Market problem, previous results, and our contribution. In Section 4.3 we give a formal description of the model.

4.2 The Housing Market Problem

Shapley and Scarf [SS74] introduced an economic model called a *housing market* – it consists of a finite set of agents and a finite set of houses. Each agent owns one house, considers some other houses acceptable and orders them according to their desirability. By the means of suitable trading, each agent wants to get the most desired house. Shapley and Scarf [SS74] consider the notion of the *economic equilibrium* in such markets – it represents the following idea. Houses are assigned prices; at these prices, each agent "sells" his house and with the received amount of money he "buys" a house. The set of prices together with the

associated trading is called an economic equilibrium if each agent gets the most preferred house that he can afford.

Shapley and Scarf [SS74] proved the existence of an economic equilibrium in all housing markets. A polynomial-time algorithm for finding an economic equilibrium, called the Top Trading Cycles algorithm, was attributed to Gale (see [SS74]). An asymptotically optimal version of the Top Trading Cycles algorithm was proposed by Abraham et al. [ACMM05].

However, the above results rely substantially on the assumption that each agent's house is unique. Fekete et al. [FSW03] proposed a modified version of the basic model: some houses are of the same type (and so must have equal price) and agents are indifferent between houses of the same type (hence, such houses are tied in all agents' preference lists). This is called a market with *duplicate houses*. In this model it may happen that the economic equilibrium does not exist. Fekete et al. even proved that the problem to decide its existence is NP-complete [FSW03].

In the market constructed in the NP-completeness reduction of Fekete et al. [FSW03], there are some ties between houses of different types in the preference lists. If no such ties are present, a polynomial-time algorithm for finding an equilibrium was proposed by Cechlárová and Fleiner [CF10]. Their algorithm represents the housing market by a directed graph whose vertices correspond to house types and arcs to agents and it performs a repeated search for strongly connected components of certain subgraphs. Hence, without further improvements its complexity is $O(NM)$, where N is the number of agents and M is the number of house types. In Chapter 5 we show that using an extension of Tarjan's depth-first search algorithm for strongly connected components of a digraph an $O(L)$ implementation can be obtained, with L denoting the total length of preference lists of all agents. This is asymptotically optimal, since the size of the input is $\Theta(L)$.

On the other hand, Cechlárová and Fleiner [CF10] showed that the problem remains NP-complete even if each agent distinguishes only three classes of house types: desired houses, houses of the same type as his original house and unacceptable houses. They call such preferences *trichotomic*.

In markets where each agent may own several units of each good it has been known for many years that equilibrium might not exist. A result of Deng, Papadimitriou and Safra states that even in the case with linear utility functions the problem to decide its existence is NP-complete [DPS03]. These authors studied the so-called ε-approximate equilibrium, i. e., such that the market clears approximately (at most ε units of each good remain unsold) and each agent obtains a commodity bundle such that its utility is within a factor of $(1 - \varepsilon)$ from the optimum in his budget set.

Cechlárová and Fleiner [CF10] proposed a different notion of approximate equilibrium for housing markets with duplicate houses. They studied the *deficiency of housing markets*, i. e., the minimum number of agents that cannot get a most preferred house in their budget set.

Cechlárová and Schlotter [CS10] examined deficiency from the parameterized

complexity viewpoint. They proved that the deficiency problem is NP-hard even in the case when each agent prefers only one house type to his own, and the maximum number of houses of the same type is two. They further proved this problem to be W[1]-hard when parameterized by the desired value of deficiency, and fixed-parameter tractable when parameterized by the number of distinct house types.

In Chapter 6, we focus on the approximability of the maximum number of satisfied agents, i. e., the maximum number of agents that *can* get a most preferred house in their budget set. We give a 2-approximation algorithm for the case when agents' preferences are trichotomic. Further, we show several inapproximability results: for the general setting with duplicate houses and possible ties between different house types, as well as for the special case of trichotomic preferences.

4.3 A Formal Description of the Model

Let A be a set of N *agents*, H a set of M *house types*; houses of the same type are said to be *duplicate*. The *endowment function* $\omega : A \to H$ assigns to each agent the type of the house he originally owns. (In the classical Shapley–Scarf housing market without duplicate houses [SS74], $M = N$ and ω is a bijection.) We denote by $A(h)$ the set of agents endowed with a house of type $h \in H$.

Each agent has preferences over house types in H in the form of a linearly ordered list $P(a)$, possibly with ties. Notation $h \succeq_a k$ means that agent a prefers houses of type h to houses of type k. The set of house types appearing in $P(a)$ is denoted by $H(a)$, and house types in $H(a)$ are said to be *acceptable* for a. We assume that $\omega(a)$ belongs to the least preferred acceptable house types for each agent. The remaining house types are called *unacceptable*.

In the special case of *trichotomic preferences*, each agent distinguishes only three kinds of house types: house types more preferred than the type of his own house, these are called *desired*; houses of the same type as his own house and *unacceptable* house types.

The N-tuple $\mathcal{P}$ of all preference lists is called the *preference profile*. The quadruple $\mathcal{M} = (A, H, \omega, \mathcal{P})$ is called a *housing market*.

If $A' \subseteq A$, we define $\omega(A')$ to be the set of house types owned by agents in A', i. e., $\omega(A') = \{\omega(a); a \in A'\} \subseteq H$. For $T \subseteq H$ and $A' \subseteq A$ we define $A'_A(T)$ to be the set of agents in A' who own a house whose type is in T, i. e., $A'_A(T) = \{a \in A'; \omega(a) \in T\}$. In the special case when $A' = A$ we write just $A(T)$. If $T = \{h\}$ for a $h \in H$, we write $A(h)$ instead of $A(\{h\})$.

A market $\mathcal{M}' = (A', H', \omega', \mathcal{P}')$ is a *submarket* of a market $\mathcal{M} = (A, H, \omega, \mathcal{P})$ if $A' \subseteq A$, $H' = \omega(A')$ and the endowment function ω' and preference profile $\mathcal{P}'$ are restrictions of ω and $\mathcal{P}$ to A'. We also say that $\mathcal{M}'$ is a *submarket of* $\mathcal{M}$ *induced by* A'. For a set $S \subseteq A$, let $\mathcal{M} \setminus S$ denote the submarket of $\mathcal{M}$ induced by $A \setminus S$.

Our aim for a market is to assign prices to house types and design trading consistent with prices so that each agent ends up with exactly one acceptable

house. We formalize these ideas by the following definitions.

We say that a pair $\mathcal{T} = (\pi, p)$ is a *solution for* $\mathcal{M}$ if

(*i*) $\pi : A \to A$ is a bijection such that $\omega(\pi(a)) \in H(a)$ for each $a \in A$,

(*ii*) $p : H \to \mathbb{R}$ is a *price function* such that $p(\omega(\pi(a))) \leq p(\omega(a))$ for each $a \in A$.

The bijection π partitions A into cycles of the form $(a_0, \ldots, a_{l-1})$, where $\pi(a_i) = a_{i+1}$ for all $i = 0, \ldots, l-1$ (modulo l), called *trading cycles*. We say that an agent a is *trading* in solution $\mathcal{T}$ if $a \neq \pi(a)$.

Given a price function $p : H \to \mathbb{R}^+$, the *budget set* of an agent a with respect to p is the set of house types that are *affordable by* a, i. e., $\{h \in H : p(h) \leq p(\omega(a))\}$. An agent a is *satisfied* in a solution $\mathcal{T}$ of $\mathcal{M}$ if $\omega(\pi(a))$ is among the most preferred house types in the budget set of a. (In the case of trichotomic preferences, an agent a is satisfied if and only if a is either trading or $p(h) > p(\omega(a))$ for each $h \in H(a)$ such that $h \neq \omega(a)$; in other words, of all acceptable houses a can only afford houses of the same type as his endowment). Otherwise, we say that a is *dissatisfied*. If all agents are satisfied in a solution (π, p), we say that (π, p) is an *economic equilibrium*.

A simple observation follows directly from the definitions (see also [CF10, FSW03]).

Lemma 4.1. *If $\mathcal{T} = (\pi, p)$ is a solution for a market $\mathcal{M}$ then $p(\omega(\pi(a))) = p(\omega(a))$ for each agent $a \in A$.*

To model markets, we use directed graphs, also called *digraphs*. Throughout this part of the thesis, all considered graphs are digraphs. A digraph may contain parallel arcs as well as loops.

Chapter 5

An Efficient Implementation of the Equilibrium Algorithm

5.1 The Problem HMD-Equilibrium

In this chapter, we focus on housing markets with duplicate houses when the preference lists of agents do not contain ties between different house types. We deal with the following problem.

HMD-Equilibrium
Input: A market $\mathcal{M}$ with duplicate houses (without any ties between different house types)
Output: An economic equilibrium in $\mathcal{M}$ or "there is no economic equilibrium".

As defined in Section 4.3, N denotes the number of agents in the considered market, and M denotes the number of house types. Cechlárová and Fleiner [CF10] found an $O(NM)$ algorithm for the problem HMD-Equilibrium. We show an efficient implementation of their algorithm – we improve the time complexity bound from $O(NM)$ to $O(L)$, where L is the total length of preference lists of all agents. This is asymptotically optimal.

In Section 5.2 we describe the algorithm of Cechlárová and Fleiner and some related results. In Section 5.3 we summarize the well-known Tarjan's algorithm for finding strongly connected components of a digraph, which is then used in our algorithm. In Section 5.4 we give our Algorithm HousingEquilibrium and prove its correctness.

5.2 The Algorithm of Cechlárová and Fleiner

Let $\mathcal{M} = (A, H, \omega, \mathcal{P})$ be a market such that the preference list $\mathcal{P}$ does not contain any indifferences between houses of different types. For each agent $a \in A$ and each set of house types $F \subseteq H$, let $f_F(a)$ denote the unique most preferred (we also say the *first-choice*) house type from the set F.

The algorithm of Cechlárová and Fleiner [CF10] is based on a representation of a housing market $\mathcal{M}$ as a digraph $G_{\mathcal{M}} = (V, E)$, called the *digraph of the housing market* $\mathcal{M}$, with vertices representing house types, i. e., $V = H$, and arcs representing agents in the following way: if $\omega(a) = h_i$ and $h_j = f_H(a)$ for an agent a, then there is an arc $e(a) = (h_i, h_j) \in E$.

Let $\mathcal{M}$ be a market. Cechlárová and Fleiner [CF10] proved that if an economic equilibrium (x, π, p) exists in $\mathcal{M}$ and if D is a sink SCC in $G_{\mathcal{M}}$, then

- the prices of all house types corresponding to vertices of D are equal,
- all agents corresponding to arcs of D trade among themselves, so D is an Eulerian digraph, and
- if $G_{\mathcal{M}}$ contains an arc (u, v) with $u \notin V(D)$ and $v \in V(D)$, i. e., there is an agent a who owns a house h of a type different from all the house types in D but his first-choice house type belongs to D, then this house may not be affordable for a, in other words, the price of u must be strictly smaller than the price of house types in D.

These observations lead to the following theorem [CF10]:

Theorem 5.1. *Let $\mathcal{M}$ be a housing market and let D be a sink SCC in $G_{\mathcal{M}}$. Then $\mathcal{M}$ admits an economic equilibrium if and only if D is Eulerian and if the submarket $\mathcal{M}' = \mathcal{M} \setminus E(D)$ also admits an economic equilibrium.*

We remark that Theorem 5.1 is a generalization of the result of Shapley and Scarf [SS74] for markets without duplicate houses: if there are no duplicate houses in $\mathcal{M}$, then each vertex of $G_{\mathcal{M}}$ corresponds to a unique house owned by just one agent, the out-degree of each vertex is 1 and the sink SCCs are simply cycles, so they are trivially Eulerian. Hence, an economic equilibrium always exists.

Theorem 5.1 leads to the following general algorithm for finding an economic equilibrium for a given housing market $\mathcal{M}$ (or showing that there is none): First, create the digraph $G_{\mathcal{M}}$ and take any sink SCC D in $G_{\mathcal{M}}$. If D is not Eulerian, then $\mathcal{M}$ does not admit any economic equilibrium. If D is Eulerian, then give to all houses of $V(D)$ the same price which is strictly smaller than any price assigned before; let the agents corresponding to $E(D)$ trade according to some Eulerian walk in D, and delete these house types and agents from $\mathcal{M}$. The whole procedure continues for the obtained submarket $\mathcal{M}' = \mathcal{M} \setminus E(D)$. However, $G_{\mathcal{M}'}$ is not simply a subdigraph of $G_{\mathcal{M}}$, as in the digraph corresponding to $\mathcal{M}'$ some new arcs are added. They correspond to agents whose endowments are in $\mathcal{M}'$ but their first-choice house types were in $V(D)$; for each such agent, there is a new arc in $G_{\mathcal{M}'}$ corresponding to his first-choice in $\mathcal{M}'$.

A sink SCC in $G_{\mathcal{M}}$ can be found using the algorithm of Tarjan [Tar72] which works in time $O(N)$ (N is the number of agents and also the number of arcs in $G_{\mathcal{M}}$ – for each agent, there is one arc). An Eulerian walk in the SCC can be obtained using Hierholzer's algorithm [Hie73]; the algorithm works in time linear in the size of the SCC (which can be bounded by $O(N)$, too).

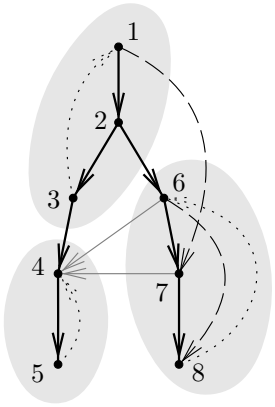


Figure 5.1: An example of a DFS forest (in this case, a tree) and strongly connected components of a digraph. Vertices are numbered by their DFS numbers.

After the deletion of a SCC, the market has lost at least one house type, so the number of iterations is bounded by $O(M)$. In the digraph corresponding to the submarket, some arcs remain from the previous iteration, but some new may emerge (as said above). If we compute the next sink SCC from scratch, we cannot ensure a better bound than $O(N)$ for the new iteration. So the complexity of the whole algorithm will be $O(NM)$.

In this chapter, we present a faster algorithm for the problem HMD-Equilibrium – using an extension of Tarjan's algorithm together with Hierholzer's algorithm, we get an algorithm that works in time $O(L)$, where L is the total length of preference lists of agents, i. e., $L = \sum_{a \in A} |H(a)|$.

5.3 An Overview of Tarjan's Algorithm

Tarjan's algorithm is a well-known algorithm for finding the list of strongly connected components of a digraph [Tar72]. In this section, we give an overview of Tarjan's algorithm and we state some known facts without proof (we refer the reader to the paper of Tarjan [Tar72] and also to the books [Gib85, TS92]). In Section 5.4 we address the details in which the algorithm HousingEquilibrium differs from Tarjan's algorithm.

Tarjan's algorithm traverses the input digraph by visiting the vertices in depth-first search (DFS) order. The digraph is thus "organized" into a rooted directed forest called the DFS forest (we refer to this DFS forest when we say that a vertex is a *root*, a *son*, a *father*, a *descendant* or an *ancestor*). The vertices are labelled by integer numbers *num* in order of their discovery by the traversal – these numbers are called their *DFS numbers*. The traversal is done by recursive calls of the procedure Visit which traverses the subtree of a given vertex.

Arcs of the digraph are classified into the following four categories at the moment of their discovery by the algorithm.

- *Forest arcs* are the arcs that lead from a vertex v to another vertex w which has not yet been discovered by the algorithm. Forest arcs are drawn as bold lines in Fig. 5.1.

- *Forward arcs* are the arcs that lead from a vertex v to a DFS descendant of v which has already been discovered by the algorithm. Forward arcs are drawn as dashed lines in Fig. 5.1.

- *Back arcs* are the arcs that lead from a vertex v to a DFS ancestor of v. Back arcs are drawn as dotted lines in Fig. 5.1.

- *Cross arcs* are the arcs that lead from a vertex v into a vertex that has already been visited, but is neither a descendant nor an ancestor of v). Cross arcs are drawn as gray lines in Fig. 5.1.

The algorithm computes for each vertex v a number $low(v)$: the number $low(v)$ is defined as the minimum of $num(w)$ for all vertices w such that w is reachable from v by a directed path consisting of some forest arcs and at most one back arc at the end of the path. The number $low(v)$ is computed for each vertex v recursively – it is initially set to $num(v)$ and it is always updated after returning from a son of v or after examining an arc leading from v.

A SCC is discovered when the algorithm finds a vertex v such that $low(v) = num(v)$. The SCC is then the subtree of v. Thus, the SCC is discovered when the computation of $low(v)$ is finished, and that happens after the algorithm has finished visiting the sons of v and examining all arcs leading from v.

The visited vertices are stored in an auxiliary stack; they are pushed on the top of the stack after their discovery. As soon as the algorithm discovers that a vertex v is a root of a SCC, all vertices from the top of the stack down to v are popped – in this way, we get exactly the vertices of the newly discovered SCC (see [TS92]). Then, the traversal proceeds.

The algorithm outputs a list of strongly connected components of the input digraph G. It works in time $O(|V| + |E|)$, where V is the set of vertices of G and E is the set of arcs of G.

5.4 The Algorithm HousingEquilibrium

The Algorithm HousingEquilibrium uses the same general ideas as the algorithm of Cechlárová and Fleiner [CF10] described in Section 5.2. The improvement is in the details how we process the digraph $G_{\mathcal{M}}$ representing the market $\mathcal{M}$ and how we maintain it after the deletion of a SCC. We address this later in this section. The algorithm constructs a list of sink SCCs and then tests whether they are Eulerian and finds Eulerian walks in them using the algorithm of Hierholzer [Hie73] (see also the book [Jun13]). If all the found SCCs are Eulerian, the algorithm outputs the equilibrium as a sequence of agents along trading cycles together with house prices.

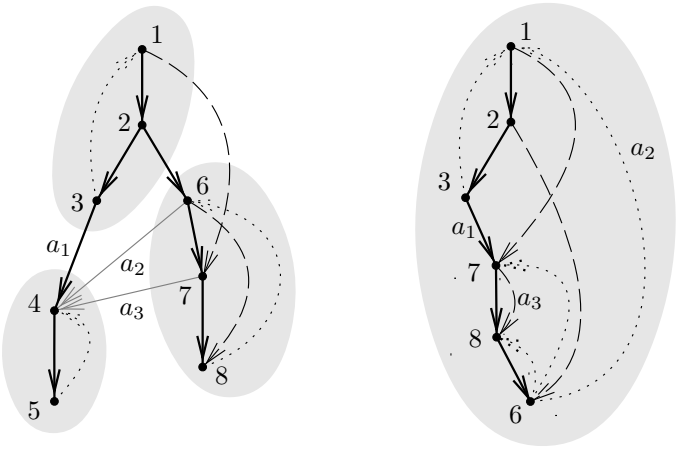


Figure 5.2: The deletion of a SCC in the Algorithm HousingEquilibrium – after the vertex with number 4 is identified as a root of a SCC and the SCC is deleted, new first-choice arcs of agents a_1, a_2 and a_3 are added and the two connected components are merged. (Note that the algorithm does not "know" the whole DFS tree at the point of deletion of the first SCC; instead, it constructs the DFS tree on the right side.)

The main part of the algorithm HousingEquilibrium is a traversal of the input digraph which is done as in Tarjan's algorithm (summarized in Section 5.3) with certain differences; the differences are described below.

We observe that we cannot simply use Tarjan's algorithm to get a list of SCCs of $G_{\mathcal{M}}$, because the removal of a SCC from $G_{\mathcal{M}}$ causes the addition of new arcs. When a SCC D is removed from the digraph, the removal of some vertices from the digraph corresponds to the removal of house types from the underlying market. These house types disappear from preference lists of agents, and some agents may thus have a different first-choice house type. Hence, we need to add the corresponding arcs to the digraph. These new arcs may cause that some strongly connected components are merged together (see Fig. 5.2).

The Algorithm HousingEquilibrium uses the following data structures. As in Tarjan's algorithm, there is a stack that stores the discovered (and not yet deleted) vertices. Further, for each vertex v there is an ordered list $Out(v)$ of the out-neighbors of v – these correspond to first-choices of agents who own a house of type v. There is another ordered list $Out'(v)$ of out-neighbors of v within the SCC containing v. For each vertex v, we also keep an indicator $stacked(v)$ so that we can test in constant time whether the vertex v is on the stack. We keep a data structure $SCCList$ which contains the found SCCs along with the prices of the corresponding house types and with the Eulerian walks. Each SCC is represented by the list of its vertices, and each Eulerian walk is represented by a list of arcs in the order in which they appear in the walk.

The Algorithm HousingEquilibrium starts by constructing the digraph $G_{\mathcal{M}}$ representing the input market $\mathcal{M}$: the set of vertices V is equal to H. For each

agent a, the house type $\omega(a)$ is the endowment of a, and $f_H(a)$ is the first-choice house type of a. The house types $\omega(a)$ and $f_H(a)$ correspond to vertices; the algorithm adds an arc $(\omega(a), f_H(a))$ into $Out(\omega(a))$. Thus, the out-neighbor lists are constructed.

The algorithm then works identically to Tarjan's algorithm on $G_{\mathcal{M}}$ until the first SCC is identified. The two differences from Tarjan's algorithm are described below.

Arc updates. After a SCC is discovered and deleted, we do not remove any arc leading to it. The algorithm continues the depth-first traversal and when it discovers an arc leading to a deleted vertex, it recognizes the arc as *invalid*. This happens when the arc leads to a vertex w such that $num(w) > 0$ and $stacked(w) = false$.

An invalid arc (v, w) is discovered either in line 7 of procedure Visit (here the SCC containing the vertex w has already been deleted before the arc (v, w) is explored) or in line 13 of procedure Visit (this happens when the recursion returns from w after w was identified as a root of a new SCC, and thus deleted). In both cases, we call the procedure Upgrade which adds an arc corresponding to the next-choice (current first-choice) house type into $Out(v)$.

Arcs within the SCC. To run the Hierholzer's algorithm to find an Eulerian walk within a SCC, we need to provide a representation of the SCC. To this end, we want to obtain for each vertex v a list $Out'(v)$ of arcs outgoing from v to vertices in the same SCC. We do the following steps (we later prove that they are sufficient):

- When the algorithm processes a new arc $e = (v, w)$, after it tests if e is valid, it adds e to $Out'(v)$.
- In line 14 of procedure Visit the algorithm removes the arc (v, w) from $Out(v)$ after the out-neighbor w of v was deleted as a root of a new SCC.

The complete algorithm is given in pseudocode in Fig. 5.3 and Fig. 5.4.

Lemma 5.2. *Let $D_1, \ldots, D_k$ be the sequence of subdigraphs that were found by the Algorithm HousingEquilibrium. Then for each $i = 1, \ldots, k$, we have that D_i is a sink strongly connected component of the digraph $G_{\mathcal{M} \setminus \bigcup_{j<i} E(D_j)}$.*

Proof. The algorithm starts with the digraph $G_{\mathcal{M}}$ and works identically to Tarjan's algorithm until the first SCC D_1 is identified. Because D_1 is a subtree of the DFS tree, it has no outgoing arcs and hence it is a sink SCC. Hence, we have proved the statement of the lemma for $i = 1$.

When the algorithm has discovered the root of D_1 and deleted D_1, the new digraph $G_{\mathcal{M} \setminus E(D_1)}$ differs from $G_{\mathcal{M}}$ in the following way: we have deleted vertices and arcs of D_1 (which correspond to some house types and agents endowed with those house types) and we have added new arcs that correspond to the new first-choices of agents. Let us denote the set of these arcs by E_1. The arcs of E_1

Algorithm HousingEquilibrium($\mathcal{M}$)
Input: A market $\mathcal{M}$ with duplicate houses (without any ties between different house types)
Output: Prices of houses in an economic equilibrium; or "there is no economic equilibrium"

$i \leftarrow 1$; {initialize the counter i}
$price \leftarrow m + 1$; {initialize the price for the next SCC}
empty the stack and $SCCList$;
$V \leftarrow H$; {the vertex set is the set of house types}
for all $v \in V$ **do**
  $num(v) \leftarrow 0$; $stacked(v) \leftarrow false$; empty $Out(v)$ and $Out'(v)$;
end for
for all $a \in A$ **do**
  add to $Out(\omega(a))$ the arc $(\omega(a), f_H(a))$ {the first-choice arc of a}
end for
while for some $u \in V$ it holds that $num(u) = 0$ **do**
  Visit(u);
end while
for all SCC in $SCCList$ **do**
  run the Hierholzer's algorithm to get an Eulerian walk in SCC;
  if SCC is not Eulerian **then**
    STOP: there is no economic equilibrium.
  end if
end for
output all the strongly connected components from $SCCList$ along with their prices and the sequences of agents from the Eulerian walks and STOP.

Procedure Upgrade(e)
Input: an arc e

let $e = (v, w)$ and let a be the agent corresponding to the arc e
if there is a next-choice house type z in $P(a)$ after w **then**
  add (v, z) into $Out(v)$
end if

Figure 5.3: The Algorithm HousingEquilibrium - part I.

```
Procedure Visit(v)
Input: a vertex v
 1: num(v) ← i; i ← i + 1;
 2: low(v) ← num(v); {initialize low(v)}
 3: push v on the top of the stack; stacked(v) ← true;
 4: while Out(v) ≠ ∅ do
 5:    take the first e ∈ Out(v); delete e from Out(v);
 6:    let e = (v, w);
 7:    if num(w) > 0 and not stacked(w) then {e is an invalid arc}
 8:       Upgrade(e);
 9:    else
10:       add e to Out'(v);
11:       if num(w) = 0 then {e is a forest arc}
12:          Visit(w);
13:          if not stacked(w) then {e became invalid}
14:             remove e from Out'(v); Upgrade(e);
15:          else
16:             low(v) ← min{low(v); low(w)};
17:          end if
18:       else
19:          if num(w) < num(v) then
20:             low(v) ← min{low(v); num(w)};
21:          end if
22:       end if
23:    end if
24: end while
25: if num(v) = low(v) then {v is the root of a new SCC}
26:    initialize the list of vertices SCC to be empty;
27:    for all u from the top of the stack down to and including v do
28:       add u to SCC;
29:       pop u from the stack; stacked(u) ← false
30:    end for
31:    add SCC along with the current value of price into SCCList;
32:    price ← price − 1;
33: end if
```

Figure 5.4: The Algorithm HousingEquilibrium - part II.

lead from vertices for which the algorithm has not yet finished (or perhaps even started) visiting, because all vertices for which the algorithm has now finished visiting have now been deleted as part of D_1. We can see that the algorithm will proceed in the same way as Tarjan's algorithm would if the arcs of E_1 were contained in the digraph from the beginning (each arc (v, w) at the end of the corresponding list $Out(v)$).

We can repeat this argument for each deletion of a SCC D_i and the arc set E_i. Hence, the claim of the lemma follows. □

Lemma 5.3. *After the traversal of the digraph, for each vertex v the list $Out'(v)$ is exactly the list of out-neighbors of v in its SCC found by the Algorithm HousingEquilibrium.*

Proof. Let v be a vertex and let D be the SCC of v. Any arc (v, w) that is contained in $Out(v)$ is added into $Out'(v)$ (line 10 of procedure Visit) unless the arc is invalid. If the arc (v, w) is invalid, then w belongs to an already deleted SCC, and hence w does not belong to D. Hence, all arcs leading from v to a vertex in D are contained in $Out'(v)$.

At the moment of deletion of D, all arcs leading from v to previously deleted SCCs have been recognized as invalid and removed from $Out'(v)$. Further, D is a sink SCC of the current digraph; hence, there are no arcs leading from D to the rest of the digraph. Hence, $Out'(v)$ contains no arcs leading to a different SCC. This finishes the proof. □

Theorem 5.4. *For a housing market $\mathcal{M}$ with duplicate houses, the Algorithm HousingEquilibrium correctly finds an economic equilibrium or decides that there is none. The algorithm works in $O(L)$ steps, where L is the total length of all preference lists of agents.*

Proof. Let $D_1, \ldots, D_k$ be the sequence of sink SCCs that were found by the Algorithm HousingEquilibrium. By Lemma 5.3 the algorithm correctly finds the arcs that belong to the SCCs.

By Lemma 5.2 and Theorem 5.1, the market $\mathcal{M}$ admits an economic equilibrium if and only if all of $D_1, \ldots, D_k$ are Eulerian. From the correctness of Hierholzer's algorithm it follows that Algorithm HousingEquilibrium correctly decides if the SCCs are Eulerian, and if they are, it finds the Eulerian walks. As in the algorithm of Cechlárová and Fleiner [CF10] described in Section 5.2, the trading relations in the resulting equilibrium are defined by the arcs along the Eulerian walks, and house prices were assigned as in the algorithm of Cechlárová and Fleiner, too. Hence, the result of the Algorithm HousingEquilibrium is correct.

We now bound the time complexity. The initialization and the construction of the input digraph from preference lists is done in time $O(M + N)$. The digraph has M vertices and at most L arcs (including those added by upgrades). Tarjan's algorithm runs in time linear in the number of vertices and arcs; hence, the complexity of Tarjan's algorithm itself (without the extra steps special for Algorithm HousingEquilibrium) on our digraph is $O(M + L)$, which is $O(L)$.

An upgrade of an arc can be done in constant time provided that we keep with each arc a pointer to the corresponding agent; for each agent we also keep a pointer to mark the last used first-choice in the preference list, so that we can get the next first-choice in constant time. Thus, the total time spent by all upgrades is $O(L)$. The total time spent by testing the invalidity of arcs, by constructing the lists Out' and by other details is $O(L)$, too.

For a SCC D_i, let d_i be the number of vertices in D_i and let n_i be the number of arcs in D_i. Hierholzer's algorithm runs in time linear in the number of vertices and arcs; hence, for each SCC, it runs in time $O(d_i + n_i)$. All the remaining manipulation with D_i (storing in $SCCList$, outputting the results) can be done in time $O(d_i+n_i)$, too. Hence, the total time for all SCCs is $O(d_1+n_1+\cdots+d_k+n_k)$, which is $O(L)$.

Altogether, the running time of the whole Algorithm HousingEquilibrium is $O(L)$. □

5.5 Concluding Remarks

We have presented an asymptotically optimal Algorithm HousingEquilibrium for computing the economic equilibrium in housing markets with duplicate houses. We remark that in the special case without duplicate houses, the algorithm may be simplified – we do not have to find Eulerian walks nor to test if the SCCs are Eulerian, because each SCC is a directed cycle. Then, the simplified algorithm HousingEquilibrium provides an asymptotically optimal implementation of the Top Trading Cycles algorithm [SS74] analogous to the implementation of Abraham et al. [ACMM05] which is used as part of an algorithm to find a maximum cardinality Pareto optimal matching.

Chapter 6

Approximability of the Economic Equilibrium

6.1 The Results of This Chapter

In this chapter, we focus on the approximability of the economic equilibrium in housing markets with duplicate houses when preference lists of agents may contain ties of different house types, and also in the special case of trichotomic preferences. For the necessary definitions, please refer to Section 4.3.

Let $\mathcal{M}$ be a market and let $\mathcal{T}$ be a solution for $\mathcal{M}$. By $\text{sat}(\mathcal{T})$ we denote the number of satisfied agents in $\mathcal{T}$ and by $\text{dissat}(\mathcal{T})$ we denote the number of dissatisfied agents in $\mathcal{T}$. By $\text{Sat}(\mathcal{T})$ and $\text{Dissat}(\mathcal{T})$ we denote the sets of satisfied and dissatisfied agents in $\mathcal{T}$, respectively. Further, by $\text{sat}(\mathcal{M})$ we denote the maximum number of satisfied agents over all solutions for $\mathcal{M}$. We study the following problems.

Max-HMDTies
Input: A market $\mathcal{M}$ with duplicate houses and such that preference lists of agents may contain ties between different house types.
Problem: Find a solution $\mathcal{T}$ that maximizes $\text{sat}(\mathcal{T})$.

Max-HMDTri
Input: A market $\mathcal{M}$ with duplicate houses and trichotomic preferences.
Problem: Find a solution $\mathcal{T}$ that maximizes $\text{sat}(\mathcal{T})$.

We prove that the problem Max-HMDTri is NP-hard to approximate within a factor smaller than 21/19, even if the maximum number of houses of the same type is two, and agents endowed with a house of the same type have the same preference lists. We further prove that the problem Max-HMDTies is NP-hard to approximate within a factor smaller than 1.2. Assuming that the Unique Games Conjecture of Khot [Kho02] is true, we obtain even stronger inapproximability results.

In Section 6.2 we study bounds for $\text{sat}(\mathcal{M})$ and we describe a 2-approximation algorithm for Max-HMDTri. In Section 6.3 we give a transformation of an

instance of the MIN-VERTEX-COVER problem to an instance of the problem MAX-HMDTIES and to an instance of the problem MAX-HMDTRI. In Section 6.4 we use the results of Section 6.3 to derive the hardness of approximation of MAX-HMDTIES and MAX-HMDTRI.

6.2 A 2-Approximation Algorithm for the Problem MAX-HMDTRI

In this section we prove that in each market with trichotomic preferences, at least half of the agents can always be satisfied. The proof of this result also provides a 2-approximation algorithm for MAX-HMDTRI.

We model a housing market $\mathcal{M}$ by a digraph $G^{\bullet}_{\mathcal{M}} = (A, E)$. Vertices of $G^{\bullet}_{\mathcal{M}}$ correspond to agents and each vertex is colored according to the type of house this agent is endowed with – hence the color classes correspond to the sets $A(h)$. An arc $ab \in E$ means that $\omega(b) \succ_a \omega(a)$.[1] In case that agents' preferences are not trichotomic, the arcs are labelled with numbers that express the preference order.

When there is no danger of confusion, we relate the properties of the digraph $G^{\bullet}_{\mathcal{M}}$ to the market $\mathcal{M}$, and we identify vertices of $G^{\bullet}_{\mathcal{M}}$ with agents of $\mathcal{M}$. We then speak of *in-neighbors* and *out-neighbors* of agents, of *directed cycles in* $\mathcal{M}$, and we say that $\mathcal{M}$ is *acyclic* if $G^{\bullet}_{\mathcal{M}}$ does not contain any directed cycle (note that in $G^{\bullet}_{\mathcal{M}}$, there are no loops and arcs between agents of the same type). For simplicity, we sometimes say a *cycle* instead of a directed cycle, and a *trichotomic market* instead of a market with trichotomic preferences.

A set of vertices in a directed graph is a called a *feedback vertex set* (FVS for short) if its removal leaves an acyclic graph. A set S of vertices of an undirected graph is called a *vertex cover* (VC for short) if each edge has at least one endvertex in S. The following problem is well known to be NP-hard; we use it in our proofs.

MIN-VERTEX-COVER
Input: An undirected graph G
Problem: Find a vertex cover of minimum size in G.

We now present several lemmas that lead towards a 2-approximation algorithm for MAX-HMDTRI (however, the lemmas themselves do not require that preference lists are trichotomic). Let $\mathcal{M}$ be a market; by L we denote the total length of the agents' preference lists, i. e., $L = \sum_{a \in A} |H(a)|$.

Lemma 6.1. *Any acyclic market $\mathcal{M}$ admits an economic equilibrium, i. e., for any acyclic market $\mathcal{M}$ with N agents we have* $\mathrm{sat}(\mathcal{M}) = N$. *Moreover, an optimal solution can be found in time $O(L)$.*

[1] In the paper [CJ11], the digraph $G^{\bullet}_{\mathcal{M}}$ is denoted just by $G_{\mathcal{M}}$. However, we use the notation $G^{\bullet}_{\mathcal{M}}$ here to avoid confusion with the digraph $G_{\mathcal{M}}$ in Chapter 5.

Proof. Let us denote by $G^*_{\mathcal{M}}$ the digraph whose vertices correspond to house types and a pair hk is an arc if and only if there exists an agent a with $\omega(a) = h$ such that $k \succ_a \omega(a)$.

We observe that if $G^\bullet_{\mathcal{M}}$ is acyclic then $G^*_{\mathcal{M}}$ is acyclic too. To see this, suppose that $(h_0, h_1, \ldots, h_{k-1})$ is a cycle in $G^*_{\mathcal{M}}$. This means that there are agents $a_0, a_1, \ldots, a_{k-1}$ such that $\omega(a_i) = h_i$ and a_i desires h_{i+1} (modulo k). It is easy to see that $(a_0, a_1, \ldots, a_{k-1})$ is a cycle in $G^\bullet_{\mathcal{M}}$.

We assign prices to house types according to any topological ordering of vertices in $G^*_{\mathcal{M}}$ so that $hk \in E(G^*_{\mathcal{M}})$ implies $p(h) < p(k)$. Such prices enable no trading, but all agents are satisfied.

Finally, the digraph $G^*_{\mathcal{M}}$ can be constructed from preference lists of agents in time $O(L)$, the number of its arcs is $O(L)$ too, hence the topological ordering of its vertices can also be performed in time $O(L)$. □

Lemma 6.2. *Let $\mathcal{M}'$ be a submarket of $\mathcal{M}$. Every solution $\mathcal{T}'$ of $\mathcal{M}'$ can be extended in time $O(L)$ to a solution $\mathcal{T}$ of $\mathcal{M}$ such that* $\mathrm{Sat}(\mathcal{T}) \supseteq \mathrm{Sat}(\mathcal{T}')$. *Hence,* $\mathrm{sat}(\mathcal{M}) \geq \mathrm{sat}(\mathcal{M}')$.

Proof. Let the trading relations in $\mathcal{T}$ be the same as in $\mathcal{T}'$; agents of $\mathcal{M}$ that are not in $\mathcal{M}'$ are not trading. All house types present in $\mathcal{M}'$ are given the same price as in $\mathcal{T}'$. House types that are only in $\mathcal{M}$ and not in $\mathcal{M}'$ are all given equal price, higher than that of any house type in $\mathcal{M}'$. This can be done in time $O(L)$.

It is straightforward to check that any agent satisfied in $\mathcal{T}'$ is satisfied in $\mathcal{T}$ as well, because trading is the same and he can afford exactly the same house types as before. Thus, $\mathrm{Sat}(\mathcal{T}) \supseteq \mathrm{Sat}(\mathcal{T}')$. □

Lemma 6.3. *Let $\mathcal{M}$ be a market. If a set $F \subseteq A$ is a FVS in $G^\bullet_{\mathcal{M}}$, then there exists a solution $\mathcal{T}$ for $\mathcal{M}$ such that $A \setminus F \subseteq \mathrm{Sat}(\mathcal{T})$; the solution $\mathcal{T}$ can be found in time $O(L)$. Hence,* $\mathrm{sat}(\mathcal{M}) \geq |A \setminus F|$.

Proof. Let $\mathcal{M}'$ be the submarket of $\mathcal{M}$ induced by $A \setminus F$. The submarket $\mathcal{M}'$ is acyclic. Hence, by Lemma 6.1 we can find in time $O(L)$ a solution $\mathcal{T}'$ for $\mathcal{M}'$ such that all agents in $\mathcal{M}'$ are satisfied. By Lemma 6.2 we can extend $\mathcal{T}'$ in time $O(L)$ to a solution $\mathcal{T}$ of $\mathcal{M}$ such that $A \setminus F \subseteq \mathrm{Sat}(\mathcal{T})$. □

For a digraph, a *maximum cycle packing* is a set of vertex-disjoint directed cycles that contains the maximum possible number of vertices. It is known that a maximum cycle packing in a digraph can be found by transforming the digraph into an instance of minimum weight perfect matching in a bipartite graph in the following way (see, for example, an analogous approach in [ABS07]). Let $G = (V, E)$ be the digraph for which we seek a maximum cycle packing. We construct a bipartite graph $G' = (V' \cup W', E')$. The sets V' and W' are two distinct copies of the set V; we denote the copies of a vertex v_i by v'_i and w'_i.

For every arc $e = (v_i, v_j)$ in E, there is an edge $\{v'_i, w'_j\}$ of weight 0 in E'. Additionally, for each vertex $v_i \in V$, there is an edge $\{v'_i, w'_i\}$ of weight 1 in E'. Then a perfect matching in G' corresponds to a cycle packing in G, and the weight

Algorithm Max-HMDTri-Approx($\mathcal{M}$)
Input: A market $\mathcal{M}$ with duplicate houses and trichotomic preferences.
Output: A solution in which at least half of the agents of $\mathcal{M}$ are satisfied.

1. Construct the graph $G^{\bullet}_{\mathcal{M}}$;
2. find a maximum cycle packing $\mathcal{C}$ of $G^{\bullet}_{\mathcal{M}}$;
3. let $A_{\mathcal{C}}$ be the set of agents contained in $\mathcal{C}$;
4. if $|A_{\mathcal{C}}| \geq N/2$, then set the price of all house types to 1, output the house prices and the cycles of $\mathcal{C}$ as trading cycles and STOP;
5. if $|A_{\mathcal{C}}| < N/2$, then
   (a) construct the submarket $\mathcal{M}'$ induced by $A \setminus A_{\mathcal{C}}$;
   (b) construct the graph $G^{*}_{\mathcal{M}'}$ and assign prices to house types in $\mathcal{M}'$ according to a topological ordering of $G^{*}_{\mathcal{M}'}$ so that $hk \in E(G^{*}_{\mathcal{M}'})$ implies $p(h) < p(k)$;
   (c) assign to the remaining house types the same price higher than the price of any house type in $\mathcal{M}'$;
   (d) output the solution with the assigned prices and no trading cycles and STOP.

Figure 6.1: The Algorithm Max-HMDTri-Approx.

of the matching is equal to the number of vertices in G that are not covered by the cycle packing.

A minimum weight perfect matching in a bipartite graph can be found using the algorithm of Gabow and Tarjan [GT89]. The algorithm works in time $O(\sqrt{n'}m' \log(n'W'))$, where n' is the number of vertices of the graph, m' is the number of edges, and W' is the largest edge weight; in our case $W' = 1$. Let n be the number of vertices of the original digraph and let m be the number of its arcs; then $n' = O(n)$ and $m' = O(m)$. Hence, a maximum cycle packing can be found in time $O(\sqrt{n}m \log(n))$.

Theorem 6.4. *In each trichotomic market $\mathcal{M}$ with N agents, at least $N/2$ agents can be satisfied. Such a solution can be found by the Algorithm Max-HMDTri-Approx which works in time $O(\sqrt{N}L \log(N))$. Hence, the Algorithm Max-HMDTri-Approx is a 2-approximation algorithm for* Max-HMDTri.

Proof. The Algorithm Max-HMDTri-Approx is given in Fig. 6.1. We prove that the algorithm is correct and that it works in the given time.

Assume that the algorithm got to Step 4 and that $|A_{\mathcal{C}}| \geq N/2$. If all houses have the same price, then all agents on trading cycles are satisfied (because in a trichotomic market, a trading agent is always satisfied). As there are $|A_{\mathcal{C}}|$ agents on trading cycles and $|A_{\mathcal{C}}| \geq N/2$, the algorithm has found a desired solution.

Now assume that the algorithm got to Step 5 and that $|A_{\mathcal{C}}| < N/2$. As $\mathcal{C}$ is a maximum cycle packing, $A_{\mathcal{C}}$ is a FVS of $\mathcal{M}$. Step (a), Step (b), and Step (c) then correspond to the procedure of Lemma 6.3. We get a solution satisfying all the agents in $A \setminus A_{\mathcal{C}}$, hence, the number of satisfied agents is at least $N/2$.

We now bound the time complexity. The number of vertices in $G^{\bullet}_{\mathcal{M}}$ is N and the number of arcs is $O(L)$. Then Step 2 to find a maximum cycle packing can be done in time $O(\sqrt{N}L\log(N))$ using the approach above with the algorithm of Gabow and Tarjan [GT89]. All other steps can be done in time $O(L)$. Hence, the time complexity of the algorithm is $O(\sqrt{N}L\log(N))$. □

The following example shows that there is no better approximation guarantee for the Algorithm Max-HMDTri-Approx than 2.

Example 6.5. *Consider the following market $\mathcal{M}$.*

$$\begin{aligned}\mathcal{M}: \qquad & A = \{a_1, a_2, \ldots, a_q, b_1, b_2, \ldots, b_q, c\}\\ & \omega(a_i) = \omega(b_i) = h_i;\ \omega(c) = h_{q+1}\\ & P(a_i) = P(b_i) = h_{i+1} \succ h_i;\ P(c) = h_1 \succ h_{q+1}\end{aligned}$$

Here, $|A| = 2q+1$ and any maximum cycle packing contains a single cycle of length $q+1$: $(c, a_1$ or b_1, a_2 or $b_2, \ldots, a_q$ or $b_q)$. Thus, the algorithm satisfies $q+1$ agents only, and since the house types of the rest of the agents are already present in the trading cycle, q agents remain unsatisfied. On the other hand, if agent c is removed, an acyclic market with $2q$ satisfied agents is obtained.

If we let q grow indefinitely, we can see that the bound for the approximation guarantee of the Algorithm Max-HMDTri-Approx cannot be tightened to $2-\varepsilon$ for any $\varepsilon > 0$.

6.3 The Reduction from MIN-VERTEX-COVER

In this section, we describe a transformation of an instance of the MIN-VERTEX-COVER problem to an instance of MAX-HMDTIES. Proposition 6.6, the main result of this section, provides a transformation of a graph G into a housing market for which the minimum number of dissatisfied agents in any optimal solution is proportional to $\mathrm{VC}(G)$.

Proposition 6.6 contains a parameter k. If $k = 1$, the constructed instance is also an instance of MAX-HMDTRI. The case when $k > 1$ is used to get stronger inapproximability bounds for the problem MAX-HMDTIES. The inapproximability results are presented in Section 6.4.

Proposition 6.6. *For every integer $k \geq 1$, there is a polynomial-time transformation $\tilde{T}_k$ from* MIN-VERTEX-COVER *to* MAX-HMDTIES *such that each graph G with $|V(G)|$ vertices is transformed into a housing market $\tilde{\mathcal{M}}_k = \tilde{T}_k(G)$ with $N = (2k+1)|V(G)|$ agents and such that $\mathrm{sat}(\tilde{\mathcal{M}}_k) = (2k+1)|V(G)| - k\,\mathrm{VC}(G)$.*

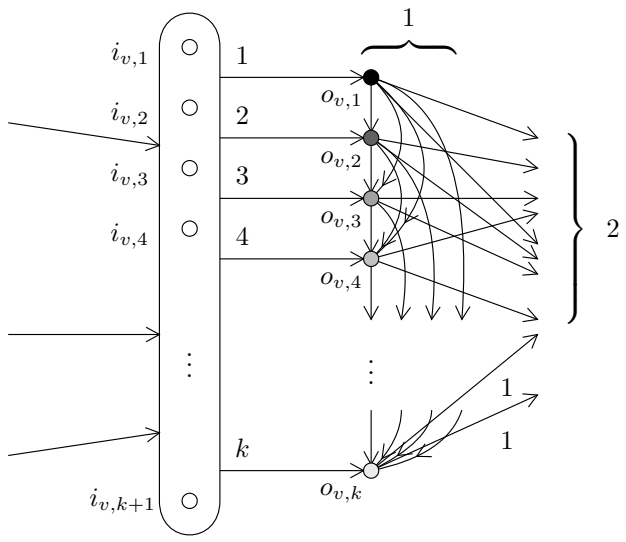


Figure 6.2: Agents in $\tilde{\mathcal{M}}_k$ corresponding to a vertex v of G'.

Proof. Consider an instance G of Min-Vertex-Cover. We construct from G a market $\tilde{\mathcal{M}}_k$.

For each vertex $v \in G$, there is a set of vertices A_v consisting of $k+1$ *incoming* agents $I_v = \{i_{v,1}, i_{v,2}, \ldots, i_{v,k+1}\}$, and k *outgoing* agents $O_v = \{o_{v,1}, o_{v,2}, \ldots, o_{v,k}\}$. Their endowments and preferences are as follows (the sets of house types in brackets in the preference lists of outgoing agents represent ties).

$$\omega(i_{v,1}) = \omega(i_{v,2}) = \cdots = \omega(i_{v,k+1}) = h_v$$
$$\omega(o_{v,1}) = h^*_{v,1};\ \omega(o_{v,2}) = h^*_{v,2};\ \ldots\ ;\ \omega(o_{v,k}) = h^*_{v,k}$$

$$P(i_{v,1}) = \cdots = P(i_{v,k+1}) = h^*_{v,1} \succ h^*_{v,2} \succ \cdots \succ h^*_{v,k} \succ h_v$$
$$P(o_{v,1}) = (h^*_{v,2}, \ldots, h^*_{v,k}) \succ (\text{all } h_w \text{ such that } \{vw\} \in E(G)) \succ h^*_{v,1}$$
$$P(o_{v,2}) = (h^*_{v,3}, \ldots, h^*_{v,k}) \succ (\text{all } h_w \text{ such that } \{vw\} \in E(G)) \succ h^*_{v,2}$$
$$\vdots$$
$$P(o_{v,k}) = (\text{all } h_w \text{ such that } \{vw\} \in E(G)) \succ h^*_{v,k}.$$

For illustration, see Fig. 6.2. Incoming (white) agents have the same in-arcs and out-arcs, which is for simplicity illustrated by just one copy of the arcs coming into and going out of the oval shape. Numbers accompanying arcs express the preference ordering.

Clearly, this construction can be performed in polynomial time.

For a vertex subset B of G, we define the corresponding set O_B of outgoing agents: $O_B = \{o_{v,j} : v \in B, j = 1, \ldots, k\}$. We prove several properties of $\tilde{\mathcal{M}}_k$.

Claim 6.7. *F is a vertex cover in G if and only if O_F is a FVS in $\tilde{\mathcal{M}}_k$.*

We observe that each cycle C in $\tilde{\mathcal{M}}_k$ has the following form: it goes through one incoming agent corresponding to v_1, then through one or more outgoing agents corresponding to v_1, through one incoming agent corresponding to v_2, one or more outgoing agents corresponding to v_2, etc. Moreover, there is always an edge $\{uv\}$ in G if there is an arc from some outgoing agent in O_u to some incoming agent I_v.

Now suppose that F is a vertex cover in G and C any cycle in $\tilde{\mathcal{M}}_k$. As said above, C corresponds to a cycle $(v_1, v_2, \ldots, v_\ell)$ in G. As F is a vertex cover in G and $\{v_1, v_2\} \in E$, we have that either $v_1 \in F$ or $v_2 \in F$. Then it is easy to see that leaving out agents in O_F will destroy C, so O_F is a FVS in $\tilde{\mathcal{M}}_k$.

To prove the converse implication we observe that for each edge $\{u, v\}$ in G, the market $\tilde{\mathcal{M}}_k$ contains a cycle crossing I_u, then O_u, then I_v, O_v and coming back to I_u. If O_F is a FVS in $\tilde{\mathcal{M}}_k$ then the set F must contain either v or u, which means that F is a vertex cover in G. This finishes the proof of Claim 6.7.

Claim 6.8. *There exists an optimal solution for $\tilde{\mathcal{M}}_k$ with no trading.*

Assume that every optimal solution has at least one trading cycle, and let $\mathcal{T}$ be an optimal solution with the minimum number of trading cycles. Let C be any trading cycle of $\mathcal{T}$. By Lemma 4.1, houses of all agents on C have the same price; we denote this price by p_C.

Now consider the set A_v for any v such that at least one agent $a \in I_v$ belongs to C. Then the price of the house of each agent from O_v who is trading must also be equal to p_C, as some of the agents preceding him on his trading cycle must belong to I_v. Thus, each agent of A_v is either trading with price p_C or not trading.

Let A_C^+ denote the union of all sets A_v such that at least one agent in A_v is trading with price p_C.

We construct another solution $\mathcal{T}'$ from $\mathcal{T}$ by modifying prices of houses according to their owner and their price in $\mathcal{T}$ as follows.

1. Owner in $A \setminus A_C^+$, price $\leq p_C$. These houses receive the same price in $\mathcal{T}'$ as in $\mathcal{T}$.
2. Owner in $A \setminus A_C^+$, price $> p_C$. These houses receive their price in $\mathcal{T}$ increased by $k + 1$.
3. Owner in A_C^+. For each v such that $A_v \subseteq A_C^+$, we set $p(h_v) = p_C$, $p(h^*_{v,1}) = p_C + 1$, $p(h^*_{v,2}) = p_C + 2$, $\ldots$, $p(h^*_{v,k}) = p_C + k$.

It is immediate that prices of the houses are defined consistently. Trading cycles from $\mathcal{T}$ with price different from p_C are in $\mathcal{T}'$ as well, while agents of A_C^+ are not trading in $\mathcal{T}'$. This concludes the construction of $\mathcal{T}'$.

Clearly, the number of trading cycles in $\mathcal{T}'$ is strictly smaller than the number of trading cycles in $\mathcal{T}$. Now we show that $\mathrm{sat}(\mathcal{T}') \geq \mathrm{sat}(\mathcal{T})$.

Let us first consider an agent outside A_C^+. His budget set in $\mathcal{T}'$ has not increased, and his trading/nontrading status has not changed compared to $\mathcal{T}$, so if this agent was satisfied in $\mathcal{T}$, he remains satisfied in $\mathcal{T}'$ as well.

Now we take a set $A_v \subseteq A_C^+$. We observe that all agents from I_v have the same strict preferences and the same nonempty budget set in $\mathcal{T}$. Hence, since at most one of them can get their common unique most preferred affordable house from O_v, at most one agent from I_v is satisfied in $\mathcal{T}$. This implies that $|\operatorname{Sat}(\mathcal{T}) \cap A_v| \leq k+1$. In $\mathcal{T}'$ no agent from A_v is trading, but since agents from I_v cannot afford any house from O_v, they are all satisfied. This implies $|\operatorname{Sat}(\mathcal{T}') \cap A_v| \geq k+1$.

To sum it up, $\operatorname{sat}(\mathcal{T}') \geq \operatorname{sat}(\mathcal{T})$. Hence, $\mathcal{T}'$ is another optimal solution having less trading cycles than $\mathcal{T}$, which is a contradiction with the choice of $\mathcal{T}$. Thus, Claim 6.8 is proved.

Claim 6.9. *Let $\mathcal{T}$ be any optimal solution of $\tilde{\mathcal{M}}_k$ with no trading and let $v \in G$ be arbitrary. Then $I_v \subseteq \operatorname{Sat}(\mathcal{T})$ and either $O_v \subseteq \operatorname{Sat}(\mathcal{T})$ or $O_v \subseteq \operatorname{Dissat}(\mathcal{T})$.*

Let $\mathcal{T}$ be an optimal solution with no trading and let v be any vertex from $V(G)$. First we claim that

$$p(h_v) < p(h^*_{v,j}) \text{ for each } j \in \{1, \ldots, k\}. \tag{6.1}$$

Otherwise, if $p(h_v) \geq p(h^*_{v,j})$ for some j, then all the agents in I_v can afford the acceptable house $h^*_{v,j}$ and are dissatisfied. By increasing $p(h^*_{v,j})$ to $p(h_v)+1$ for every such j, all $k+1$ agents in I_v become satisfied, while at most k agents from O_v may become dissatisfied. This is a contradiction with optimality of $\mathcal{T}$. Hence, the condition (6.1) holds, and this implies that $I_v \subseteq \operatorname{Sat}(\mathcal{T})$.

We now distinguish two cases.

Case 1. $p(h_v) < p(h_u)$ for each $u \in V$ such that $\{v,u\} \in E$. Let us denote $p^* = \min\{p(h_u); \{v,u\} \in E\}$ and define new prices p' fulfilling the following inequality:

$$p(h_v) < p'(h^*_{v,1}) < p'(h^*_{v,2}) < \cdots < p'(h^*_{v,k}) < p^*.$$

Thanks to the condition (6.1), the satisfaction of any agent outside A_v with respect to new prices does not differ from his satisfaction according to old prices. Moreover, all agents in O_v are satisfied. So they all were satisfied in $\mathcal{T}$ too, or $\mathcal{T}$ was not optimal.

Case 2. There exists $u \in V$ such that $\{v,u\} \in E$ and $p(h_v) \geq p(h_u)$. Due to the condition (6.1), all agents from O_v can afford the house h_u and so they are all dissatisfied, as there is no trading in $\mathcal{T}$. This finishes the proof of Claim 6.9.

To prove Proposition 6.6, we show that $\operatorname{VC}(G) = l$ if and only if $\operatorname{sat}(\tilde{\mathcal{M}}_k) = N - kl$. Assume that F is a vertex cover of size l in G. Then by Claim 6.7 the set O_F is a feedback vertex set in $\tilde{\mathcal{M}}_k$ and by Lemma 6.3, $\operatorname{sat}(\tilde{\mathcal{M}}_k) \geq |A \setminus O_F| = N - kl$.

Now let $\mathcal{T}$ be an optimal solution in $\tilde{\mathcal{M}}_k$ with no trading (which exists by Claim 6.8). Claim 6.9 implies that $\operatorname{Dissat}(\mathcal{T}) = \bigcup_{v \in B} O_v$ for some $B \subseteq V$, and hence $\operatorname{dissat}(\mathcal{T}) = kl$ for some l. For any directed cycle C in $\tilde{\mathcal{M}}_k$, at least one agent will have maximum price among the agents of C, and because there is no trading, this agent is dissatisfied. Moreover, the condition (6.1) implies that the dissatisfied agent is outgoing. Hence, the set O_B contains at least one agent out

of every directed cycle in $\tilde{\mathcal{M}}_k$, and thus O_B is a FVS of $\tilde{\mathcal{M}}_k$. Claim 6.7 directly implies that B is a vertex cover in G and hence $\mathrm{VC}(G) \leq l$.

Therefore $\mathrm{sat}(\tilde{\mathcal{M}}_k) = N - kl$ if and only if $\mathrm{VC}(G) = l$. □

In the case that $k = 1$, the constructed market $\tilde{\mathcal{M}}_k$ is trichotomic. Hence, we get the following proposition as a corollary.

Proposition 6.10. *There is a polynomial-time transformation T from* Min-Vertex-Cover *to* Max-HMDTri *such that each graph G with $|V(G)|$ vertices is transformed into a housing market $\mathcal{M} = T(G)$ with $N = 3|V(G)|$ agents, such that* $\mathrm{sat}(\mathcal{M}) = 3|V(G)| - \mathrm{VC}(G)$.

6.4 Hardness of Approximation of Max-HMDTri and Max-HMDTies

In this section we use the results of Section 6.3 to prove the hardness of approximation of the problems Max-HMDTri and Max-HMDTies.

Halldórsson et al. [HIMY07], when studying the approximability of the Stable Marriage problem with incomplete lists and ties (SMTI for short), presented a construction that assigns to each graph $G = (V, E)$ an SMTI instance I such that the number of men as well as the number of women in I is equal to $3|V(G)|$, and $|\mathrm{opt}(I)| = 3|V(G)| - \mathrm{VC}(G)$, where $\mathrm{opt}(I)$ is the size of the optimum solution for the instance I. Since the quantitative relations in our Proposition 6.10 and in the proof of their Theorem 3.2. are the same, we get by exactly the same argument the following analogies of Theorem 3.2., its Corollary 3.4. and Remark 3.6. of [HIMY07].

Proposition 6.11. *For any $\varepsilon > 0$ and $p < \frac{3-\sqrt{5}}{2}$, given an instance $\mathcal{M}$ of* Max-HMDTri *with N agents, it is NP-hard to distinguish between the following two cases:*

1. $\mathrm{sat}(\mathcal{M}) \geq \frac{2+p-\varepsilon}{3} N$, *and*
2. $\mathrm{sat}(\mathcal{M}) < \frac{2+\max\{p^2, 4p^3-3p^4\}+\varepsilon}{3} N$.

Theorem 6.12. *It is NP-hard to approximate* Max-HMDTri *within a factor smaller than* 21/19, *even if the maximum number of houses of the same type is two, and agents endowed with a house of the same type have the same preference lists.*

Remark 6.13. If Min-Vertex-Cover is NP-hard to approximate within a factor of $2 - \varepsilon$ then Max-HMDTri is NP-hard to approximate within a factor smaller than 1.25.

We now explain the assumption of Remark 6.13. In 2002, Khot [Kho02] proposed the *Unique Games Conjecture* (UGC). It states that a specific computational problem called the Unique Game is hard to approximate. For the exact

statement of the conjecture and all necessary definitions, see [Kho02, KR08]. If UGC is true, it implies the hardness of approximation of other problems. Khot and Regev [KR08] proved that if UGC is true, then Min-Vertex-Cover is NP-hard to approximate within a ratio smaller than 2. Thus, the assumption of Remark 6.13 would be fulfilled. We get the following corollary.

Corollary 6.14. *If UGC is true, then* Max-HMDTri *is NP-hard to approximate within a factor smaller than* 1.25*.*

The results of Austrin et al. [AKS09, page 3 and Theorem 4.1] may be stated in the following way.

Proposition 6.15. *If UGC is true, then for every sufficiently large integer d it is NP-hard to distinguish between the following two cases:*

1. $\mathrm{VC}(G) \leq (1/2 + \Theta(\frac{\log\log d}{\log d}))|V(G)|$, *and*
2. $\mathrm{VC}(G) > (1 - \frac{1}{\log d})|V(G)|$,

even on graphs of maximum degree d.

It is easy to see that in the transformation of Proposition 6.10, the resulting preference list lengths are bounded in terms of degrees of G. We therefore define a restricted variant of the Max-HMDTri problem with preference lists of length at most d; we call it Max-HMDTri$_d$. From Proposition 6.15 we derive the following result.

Theorem 6.16. *If UGC is true, then for every sufficiently large integer d it is NP-hard to approximate* Max-HMDTri$_d$ *within a ratio* $1.25 - \Theta(\frac{\log\log d}{\log d})$.

Proof. To a Min-Vertex-Cover instance G with maximum degree d we apply the transformation of Proposition 6.10 and obtain a market $\mathcal{M}$ with $3|V(G)|$ agents and with preference lists of length at most d (not counting the agents' own house types). By Proposition 6.10, it holds that $\mathrm{sat}(\mathcal{M}) = 3|V(G)| - \mathrm{VC}(G)$, and the two cases of Proposition 6.15 are, in the case of G, equivalent to the following.

1. $\mathrm{sat}(\mathcal{M}) \geq \frac{5}{6}N - \frac{1}{3}\Theta(\frac{\log\log d}{\log d})N$, and
2. $\mathrm{sat}(\mathcal{M}) < \frac{2}{3}N + \frac{1}{3\log d}N$.

Assume that UGC is true. Then, by Proposition 6.15, it is also NP-hard to distinguish the two cases above, which could be done by an approximation algorithm with ratio

$$\frac{\frac{5}{6}N - \frac{1}{3}\Theta\left(\frac{\log\log d}{\log d}\right)N}{\frac{2}{3}N + \frac{1}{3\log d}N} =$$

$$= \frac{\frac{5}{4} + \frac{5}{4}\cdot\frac{1}{2\log d}}{1 + \frac{1}{2\log d}} - \frac{\frac{5}{4}\cdot\frac{1}{2\log d}}{1 + \frac{1}{2\log d}} - \frac{\frac{1}{2}\Theta\left(\frac{\log\log d}{\log d}\right)}{1 + \frac{1}{2\log d}} =$$

$$= \frac{5}{4} - \Theta\left(\frac{1}{\log d}\right) - \Theta\left(\frac{\log\log d}{\log d}\right) = 1.25 - \Theta\left(\frac{\log\log d}{\log d}\right).$$

□

As the problem MAX-HMDTRI is a restricted version of MAX-HMDTIES, all inapproximability results for MAX-HMDTRI apply to MAX-HMDTIES as well. However, due to the general Proposition 6.6, we obtain even stronger inapproximability results here.

Theorem 6.17. *The problem* MAX-HMDTIES *is*

1. *NP-hard to approximate within a factor smaller than* 1.2, *and*
2. *NP-hard to approximate within a factor smaller than* 1.5, *if UGC is true.*

To prove the first part of the theorem, we use the following assertion for MIN-VERTEX-COVER due to Dinur and Safra [DS05] (as in the reasoning that leads to Proposition 6.11).

Proposition 6.18. *For any* $\varepsilon > 0$ *and* $p < \frac{3-\sqrt{5}}{2}$, *given a graph* G, *it is NP-hard to distinguish between the following two cases:*

1. $\mathrm{VC}(G) \leq (1 - p + \varepsilon)|V(G)|$, *and*
2. $\mathrm{VC}(G) > (1 - \max\{p^2, 4p^3 - 3p^4\} - \varepsilon)|V(G)|$.

To prove the second part, we use the following special case of the results of Khot and Regev [KR08, Section 4].

Proposition 6.19. *If UGC is true, then for any* $\varepsilon > 0$, *given a graph* G, *it is NP-hard to distinguish between the following two cases:*

1. $\mathrm{VC}(G) \leq (\frac{1}{2} + \varepsilon)|V(G)|$, *and*
2. $\mathrm{VC}(G) > (1 - \varepsilon)|V(G)|$.

Proof of Theorem 6.17. We apply the transformation $\tilde{T}_k$ of Proposition 6.6 to a MIN-VERTEX-COVER instance G, and obtain a market $\tilde{\mathcal{M}}_k$ with $N = (2k + 1)|V(G)|$ agents. By Proposition 6.18 with $p = 1/3$, for any $\varepsilon > 0$ it is NP-hard to distinguish between the following cases:

1. $\mathrm{sat}(\tilde{\mathcal{M}}_k) \geq \frac{\frac{4}{3}k+1-\varepsilon k}{2k+1} N$, and
2. $\mathrm{sat}(\tilde{\mathcal{M}}_k) < \frac{\frac{10}{9}k+1+\varepsilon k}{2k+1} N$.

Consider an approximation algorithm for MAX-HMDTIES with approximation ratio $1.2 - \delta$, which is equal to $\frac{6}{5} - \delta$. In the first case the algorithm gives

a solution with at least $\frac{1}{(6/5)-\delta}\frac{\frac{4}{3}k+1-\varepsilon k}{2k+1}N$ satisfied agents. If we choose k large enough, there exists a positive ε such that $\varepsilon < \frac{50\delta k+45\delta-9}{99k-45\delta k}$. Then

$$\frac{6}{5}-\delta < \frac{\frac{4}{3}k+1-\varepsilon k}{\frac{10}{9}k+1+\varepsilon k},$$

and the number of satisfied agents is larger than $\frac{\frac{10}{9}k+1+\varepsilon k}{2k+1}N$. In the second case, however, the approximation algorithm finds a solution with less than $\frac{\frac{10}{9}k+1+\varepsilon k}{2k+1}N$ satisfied agents, and hence, it can distinguish between the first and the second case.

It remains to prove the second part of the theorem. By using the cases of Proposition 6.19 for a market $\tilde{\mathcal{M}}_k$, and assuming that UGC is true, we get that it is NP-hard to distinguish

1. $\text{sat}(\tilde{\mathcal{M}}_k) \geq \frac{\frac{3}{2}k+1+\varepsilon k}{2k+1}N$, and
2. $\text{sat}(\tilde{\mathcal{M}}_k) < \frac{k+1-\varepsilon k}{2k+1}N$.

Analogously to the reasoning above, we get the desired result for a large enough k and for ε small enough. □

6.5 Concluding Remarks

We have presented a simple 2-approximation algorithm for the problem Max-HMDTri. An approximation algorithm for Max-HMDTies is not known.

Problem 6.20. Is there an approximation algorithm for Max-HMDTies?

We may also address the question of markets $\mathcal{M}$ for which $\text{sat}(\mathcal{M})$ is low. The following observation gives a simple example.

Observation 6.21. *Suppose that a housing market $\mathcal{M}$ fulfills that $|H| = 2$ and $H(a) \setminus \{\omega(a)\} \neq \emptyset$ for each agent. Then* $\text{sat}(\mathcal{M}) = \max\left(2\min\left(N_1, N_2\right), N_1, N_2\right)$, *where $|A(h_1)| = N_1$, and $|A(h_2)| = N_2$.*

Proof. In this case, $G^{\bullet}_{\mathcal{M}}$ is a complete bipartite digraph. For any solution $\mathcal{T}$, there are three possibilities. If $p(h_1) = p(h_2)$ then each trading cycle is even and contains alternately agents from $A(h_1)$ and $A(h_2)$, hence the same number of agents of the two types. Hence, $\text{sat}(\mathcal{T}) = 2\min\{N_1, N_2\}$. If $p(h_1) < p(h_2)$ then there is no trading, but all the agents from $A(h_1)$ are satisfied, as they cannot afford a house of type h_2. Thus, $\text{sat}(\mathcal{T}) = N_1$. Finally, if $p(h_1) > p(h_2)$ then $\text{sat}(\mathcal{T}) = N_2$ by a similar argument. □

Let $N_2 = 2N_1$, and let $N = N_1 + N_2$. Then observation 6.21 gives an infinite number of housing markets with $\text{sat}(\mathcal{M}) = \frac{2}{3}N$. However, it is not known whether there is a housing market with $\text{sat}(\mathcal{M})$ lower than $\frac{2}{3}N$.

Problem 6.22. Is there a housing market $\mathcal{M}$ with duplicate houses and possible ties in the agents' preference lists, such that there are N agents in $\mathcal{M}$ and $\text{sat}(\mathcal{M}) < \frac{2}{3}N$?

Bibliography

[ABS07] David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, EC '07, pages 295–304, New York, USA, 2007. ACM.

[ACKM06] David J. Abraham, Ning Chen, Vijay Kumar, and Vahab S. Mirrokni. Assignment problems in rental markets. In Paul Spirakis, Marios Mavronicolas, and Spyros Kontogiannis, editors, *Internet and Network Economics*, volume 4286 of *Lecture Notes in Computer Science*, pages 198–213. Springer Berlin Heidelberg, 2006.

[ACMM05] David J. Abraham, Katarína Cechlárová, David F. Manlove, and Kurt Mehlhorn. Pareto optimality in House Allocation problems. In Rudolf Fleischer and Gerhard Trippen, editors, *Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 3–15. Springer Berlin Heidelberg, 2005.

[AKS09] Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity*, CCC '09, pages 74–80, Washington, DC, USA, 2009. IEEE Computer Society.

[APR09] Atila Abdulkadiroğlu, Parag A. Pathak, and Alvin E. Roth. Strategy-proofness versus efficiency in matching with indifferences: Redesigning the New York City high school match. Working Paper 14864, National Bureau of Economic Research, April 2009.

[BCLS87] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[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.

[Cam77] Peter J. Cameron. Cohomological aspects of two-graphs. *Mathematische Zeitschrift*, 157(2):101–119, 1977.

[CAR] Canadian Resident Matching Service website. `http://www.carms.ca`. Accessed 19 March 2016.

[CC80] Charles J. Colbourn and Derek G. Corneil. On deciding switching equivalence of graphs. *Discrete Applied Mathematics*, 2(3):181–184, 1980.

[CF10] Katarína Cechlárová and Tamás Fleiner. Housing markets through graphs. *Algorithmica*, 58(1):19–33, 2010.

[CJ11] Katarína Cechlárová and Eva Jelínková. Approximability of economic equilibrium for housing markets with duplicate houses. In Petr Kolman and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science*, volume 6986 of *Lecture Notes in Computer Science*, pages 95–106. Springer Berlin Heidelberg, 2011.

[CS02] Yan Chen and Tayfun Sönmez. Improving efficiency of on-campus housing: An experimental study. *American Economic Review*, 92(5):1669–1686, 2002.

[CS10] Katarína Cechlárová and Ildikó Schlotter. Computing the deficiency of housing markets with duplicate houses. In Venkatesh Raman and Saket Saurabh, editors, *Parameterized and Exact Computation*, volume 6478 of *Lecture Notes in Computer Science*, pages 72–83. Springer Berlin Heidelberg, 2010.

[CT04] Peter J. Cameron and Sam Tarzi. Switching with more than two colours. *European Journal of Combinatorics*, 25(2):169–177, 2004. In memory of Jaap Seidel.

[CWJ86] Peter J. Cameron and Albert L. Wells Jr. Signatures and signed switching classes. *Journal of Combinatorial Theory, Series B*, 40(3):344–361, 1986.

[Din70] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math Doklady*, 11:1277–1280, 1970.

[dMR05] F. de Montgolfier and M. Rao. The bi-join decomposition. *Electronic Notes in Discrete Mathematics*, 22:173–177, 2005.

[Doo79] Michael Doob. Seidel switching and cospectral graphs with four distinct eigenvalues. *Annals of the New York Academy of Sciences*, 319(1):164–168, 1979.

[DPS03] Xiaotie Deng, Christos Papadimitriou, and Samuel Safra. On the complexity of price equilibria. *Journal of Computer and System Sciences*, 67(2):311–324, 2003. Special Issue on STOC 2002.

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[EHHR00a] Andrzej Ehrenfeucht, Jurriaan Hage, Tero Harju, and Grzegorz Rozenberg. Complexity issues in switching of graphs. In Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Theory and Application to Graph Transformations*, volume 1764 of *LNCS*, pages 59–70. Springer, Heidelberg, 2000.

[EHHR00b] Andrzej Ehrenfeucht, Jurriaan Hage, Tero Harju, and Grzegorz Rozenberg. Pancyclicity in switching classes. *Information Processing Letters*, 73(5-6):153–156, 2000.

[EHHR04] Andrzej Ehrenfeucht, Jurriaan Hage, Tero Harju, and Grzegorz Rozenberg. Embedding in switching classes with skew gains. In Hartmut Ehrig, Gregor Engels, Francesco Parisi-Presicce, and Grzegorz Rozenberg, editors, *Graph Transformations: Second International Conference, ICGT 2004, Rome, Italy*, pages 257–270. Springer Berlin Heidelberg, 2004.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[FF10] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

[Fra64] W. D. Frazer. A graph theoretic approach to linear codes. *Presented at the Allerton Conference, University of Illinois, Urbana*, 1964.

[FSW03] Sándor P. Fekete, Martin Skutella, and Gerhard J. Woeginger. The complexity of economic equilibria for house allocation markets. *Information Processing Letters*, 88(5):219–223, 2003.

[GI89] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989.

[Gib85] A. M. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[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.

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[GKK+10] Naveen Garg, Telikepalli Kavitha, Amit Kumar, Kurt Mehlhorn, and Julián Mestre. Assigning papers to referees. *Algorithmica*, 58(1):119–136, 2010.

[GR97] A. V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science 1997*, pages 2–11, October 1997.

[GS62] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[GT88] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, October 1988. An extended abstract appeared at STOC 86.

[GT89] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[GY05] Jonathan L. Gross and Jay Yellen. *Graph Theory and Its Applications*. CRC Press, 2nd edition, 2005.

[Hag01] Jurriaan Hage. *Structural Aspects Of Switching Classes*. PhD thesis, Leiden Institute of Advanced Computer Science, 2001.

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

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

[Her06] Jan Herman. Computational complexity in graph theory. Master's thesis, Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University in Prague, 2006.

[HF65] S. Hakimi and H. Frank. Cut-set matrices and linear codes. *IEEE Transactions on Information Theory*, 11(3):457–458, 1965.

[HH00] Jurriaan Hage and Tero Harju. The size of switching classes with skew gains. *Discrete Mathematics*, 215(1-3):81–92, 2000.

[HH04] Jurriaan Hage and Tero Harju. A characterization of acyclic switching classes of graphs using forbidden subgraphs. *SIAM J. Discrete Math.*, 18(1):159–176, 2004.

[HHW02] Jurriaan Hage, Tero Harju, and Emo 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.

[Hie73] Carl Hierholzer. Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechnung zu umfahren. *Math. Ann.*, 6:30–42, 1873.

[HIMY07] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. Improved approximation results for the stable marriage problem. *ACM Trans. Algorithms*, 3(3), 2007.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[Huf64] D. A. Huffman. A graph-theoretic formulation of binary group codes. *Summaries of papers presented at ICMCI, part 3*, pages 29–30, 1964.

[HZ79] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979.

[JJK] Vít Jelínek, Eva Jelínková, and Jan Kratochvíl. On the hardness of switching to a small number of edges. Submitted.

[JK14] Eva Jelínková and Jan Kratochvíl. On switching to H-free graphs. *Journal of Graph Theory*, 75(4):387–405, 2014.

[JRM] Japan Residency Matching Program website. `http://www.jrmp.jp`. Accessed 19 March 2016.

[JSHK11] Eva Jelínková, Ondřej Suchý, Petr Hliněný, and Jan Kratochvíl. Parameterized problems related to Seidel's switching. *Discrete Mathematics and Theoretical Computer Science*, 13(2):19–42, 2011.

[Jun13] Dieter Jungnickel. *Graphs, Networks and Algorithms*, volume 5 of *Algorithms and Computation in Mathematics*. Springer-Verlag Berlin Heidelberg, 4th edition, 2013.

[Kas61] T. Kasami. A topological approach to construction of group codes. *J. Inst. Elec. Communication Engrs. (Japan)*, 44:1316–1321, 1961.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[KNZ92] Jan Kratochvíl, Jaroslav Nešetřil, and Ondřej 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.

[Koz15] Sergiy Kozerenko. On graphs with maximum size in their switching classes. *Commentationes Mathematicae Universitatis Carolinae*, 56(1):51–61, 2015.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[Kra89] Jan Kratochvíl. Perfect codes and two-graphs. *Commentationes Mathematicae Universitatis Carolinae*, 30:755–760, 1989.

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

[Lin15] Nathan Lindzey. Speeding up graph algorithms via switching classes. In Kratochvíl Jan, Mirka Miller, and Dalibor Froncek, editors, *Combinatorial Algorithms: 25th International Workshop, IWOCA 2014, Duluth, MN, USA, October 15-17, 2014, Revised Selected Papers*, pages 238–249, Cham, 2015. Springer International Publishing.

[Man13] David F. Manlove. *Algorithmics of Matching Under Preferences*. Series on Theoretical Computer Science: Volume 2. World Scientific Publishing Co. Pte. Ltd., Singapore, 2013.

[MO12] David F. Manlove and Gregg O'Malley. Paired and altruistic kidney donation in the UK: Algorithms and experimentation. In Ralf Klasing, editor, *Experimental Algorithms*, volume 7276 of *Lecture Notes in Computer Science*, pages 271–282. Springer Berlin Heidelberg, 2012.

[MV80] S. Micali and Vijay V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27, Oct 1980.

[MW14] Jiří Matoušek and Uli Wagner. On Gromov's method of selecting heavily covered points. *Discrete Comput. Geom.*, 52(1):1–33, July 2014.

[NOB] The official website of the nobel prize. `http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2012/press.html`. Accessed 19 March 2016.

[NRM] National Resident Matching Program website. `http://www.nrmp.org`. Accessed 19 March 2016.

[Ond06] Eva Ondráčková. Computational complexity in graph theory. Master's thesis, Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University in Prague, 2006.

[PPR08] Nitsan Perach, Julia Polak, and Uriel G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion. *International Journal of Game Theory*, 36(3-4):519–535, 2008.

[QW04] Thomas Quint and Jun Wako. On houseswapping, the strict core, segmentation, and linear programming. *Mathematics of Operations Research*, 29(4):861–877, 2004.

[Rot82] Alvin E. Roth. Incentive compatibility in a market with indivisible goods. *Economics Letters*, 9(2):127–132, 1982.

[RP77] Alvin E. Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977.

[RS90] Alvin E. Roth and Marilda A. Oliveira Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis.* Econometric Society Monographs, Vol. 18. Cambridge University Press, Cambridge, UK, 1990.

[RSÜ04] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, 2004.

[RU82] D. Rotem and J. Urrutia. Circular permutation graphs. *Networks*, 12:429–437, 1982.

[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.

[Sei92] J. J. Seidel. More about two-graphs. In *Combinatorics, graphs and complexity, Proc. 4th Czech. Symp., Prachatice 1990*, volume 51 of *Annals of Discrete Math.*, pages 297–308. North Holland, Amsterdam, 1992.

[Sri96] R. Sritharan. A linear time algorithm to recognize circular permutation graphs. *Networks*, 27:171–174, 1996.

[SS74] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

[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.

[SÜ05] Tayfun Sönmez and M. Utku Ünver. House allocation with existing tenants: an equivalence. *Games and Economic Behavior*, 52(1):153–185, 2005.

[Tar72] Robert Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[TS92] K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms*. Wiley, 1992.

[Wak84] Jun Wako. A note on the strong core of a market with indivisible goods. *Journal of Mathematical Economics*, 13(2):189–194, 1984.

[Wel84] Albert L Wells. Even signings, signed switching classes, and (-1,1)-matrices. *Journal of Combinatorial Theory, Series B*, 36(2):194–212, 1984.

[Yua96] Yufei Yuan. Residence exchange wanted: A stable residence exchange problem. *European Journal of Operational Research*, 90(3):536–546, 1996.

[Zho90] Lin Zhou. On a conjecture by Gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, October 1990.

List of Abbreviations

CNF	conjunctive normal form
DFS	depth-first search
DVD	digital video disc
FVS	feedback vertex set
HMD	housing market with duplicate houses
NP	the class of decision problems that are solvable by a non-deterministic Turing machine in polynomial time
SAT	satisfiability
SCC	strongly connected component
SMTI	the Stable Marriage problem with ties and incomplete lists
UGC	the Unique Games Conjecture
VC	vertex cover